TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: N2612 - Elektrotechnika a informatika
Studijní obor: 3906T001 - Mechatronika

Řízení výkonové pasivní zátěže pomocí PC
-
Electronic control of power passive load using PC

DIPLOMOVÁ PRÁCE

Autor: **Bc. Milan Syrový**
Vedoucí práce: Ing. Martin Černík, Ph.D
Konzultant: Ing. Jan Václavík

V Liberci 17. 5. 2011

Zadání

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 - školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce.

V Liberci ……

……
Bc. Milan Syrový

Poděkování

Touto cestou bych chtěl poděkovat vedoucímu mé diplomové práce Ing. Martinovi Černíkovi, Ph. D. za hodnotné rady a odborné vedení během mé práce. Velký dík také patří doc. Ing. Milanovi Kolářovi, CSc. za konzultace týkající se programovatelných hradlových polí.

Abstrakt

Diplomová práce se zabývá návrhem a realizací řídicího obvodu výkonové pasivní zátěže a jejím ovládáním pomocí PC. Zařízení bylo vyvinuto pro laboratoř elektrických pohonů.

Požadovanou hodnotu elektrického odporu zařízení je možné volit na ovládacím panelu zařízení, nebo pomocí PC přes sériovou sběrnici USB.

V teoretické části práce jsou vysvětleny a popsány možnosti implementace USB a rozbor tohoto rozhraní. Podrobně jsou popsány možnosti využití USB převodníků FTDI a jejich knihoven v běžných vývojových programovacích a výpočetních platformách.

Praktická část pojednává o realizaci zařízení, které je postaveno na základě integrovaných obvodů FPGA Altera EPM7128 a USB převodníku FT245BL. Je proveden rozbor VHDL programu, implementovaného do FPGA, jenž řídí celé zařízení. Taktéž je provedena analýza aplikace vytvořené v prostředí LabVIEW, která vytváří uživatelské rozhraní na PC.

Na závěr se provádí ověření funkce a parametrů realizovaného zařízení. Přílohou této práce je jednoduchý návod k obsluze a instalaci zařízení, kompletní dokumentace, ovladače a knihovny, potřebné k pozdější výrobě nebo modifikaci zařízení.

Klíčová slova: výkonová odporová zátěž

zhášení elektrického oblouku

programovatelné hradlové pole

USB převodník FTDI

VHDL

Abstract

This thesis deals with the design and the implementation of an electronic control circuit of the power passive load and control using a PC. The device is being developed for the laboratory of electrical drives.

The desired value of the electrical resistance of the device can be selected on the control panel, or by use of a PC which is connected to the USB serial bus.

In the theoretical part is explained and discussed the possibility of implementing USB and the analysis of the interface. It is described the possibility of using the USB FTDI converters and their libraries in the normal development of programming and computing platforms.

The practical part deals with the implementation of the device, which is built on the basis of integrated circuits FPGA Altera EPM7128 and FT245BL USB converter. There is an analysis of the VHDL program implemented in the FPGA, which controls the entire system. It also analyzed the application which is programmed in the LabVIEW, which creates a user interface on the PC.

In the conclusion of the verification function is performed and the parameters of the realized devices. Attachments to this work are simple operating instructions and installation, complete documentation, drivers and libraries needed for the subsequent production or modification of the device.

Keywords: power resistor load

arc quenching

field programable gate array

USB converter FTDI

VHDL

Obsah

Úvod

Výkonové rezistory, nebo reostaty jsou součástí všech laboratoří pracujících s výkonovou elektronikou. S rozvojem nových technologií roste potřeba na dálku řízených odporových zátěží, zejména pak pomocí počítače. Stejná potřeba vznikla v laboratoři elektrických pohonů na Ústavu mechatroniky a technické informatiky naší univerzity. Proto se v této práci zabývám vývojem zařízení, které umožňuje měnit vstupní impedanci nejen pomocí řídicího panelu na přístroji, ale také počítačem přes sériovou sběrnici USB.

Takovéto zařízení může sloužit například jako brzdový odpor motorů, ochranný odpor generátorů a transformátorů, zatěžovací a regulační odpor pro testování střídavých a stejnosměrných zdrojů a transformátorů.

Na současném trhu existuje řada zařízení splňující tyto požadavky. Avšak jejich cena se pohybuje řádově v desetitisících korun. A to je hlavním důvodem vzniku této práce. Vytvořit obdobné zařízení, jehož pořizovací náklady budou výrazně nižší.

Nabízená zařízení mohou pracovat na dvou principech. První možností je využití polovodičových měničů a pulzně šířkové modulace, pomocí kterých lze omezovat výstupní parametry zařízení. Tento princip však klade velké nároky na řídicí a spínací obvody takovýchto zařízení. Oproti tomu druhá varianta konstrukce je velice jednoduchá. Využívá stabilní výkonové rezistory s nízkou teplotní závislostí odporu.
Ty jsou připínány pomocí relé do odporové sítě k výstupním svorkám zařízení. Právě na principu paralelní odporové sítě výkonových drátových rezistorů je založen přístroj, který v mé diplomové práci vyvíjím. Výkonové rezistory a skříň, ve které jsou umístěny, byla vyrobena na zakázku firmou E.S.H.&F. Production, s.r.o..

1. Vymezení podmínek návrhu

V této kapitole jsou vymezeny požadavky, které by mělo zařízení Odporová zátěž splňovat nebo se k nim minimálně přiblížit. Na základě těchto požadavků a konečných parametrů přístroje je v závěru této práce provedeno zhodnocení.

Základním požadavkem pro realizaci odporové zátěže, jak dále budu nazývat zařízení výkonové odporové zátěže, je použití stávajícího silového obvodu v podobě osmi relé se zhášecími obvody elektrického oblouku, které eliminují elektrický oblouk na silových kontaktech těchto relé. Samotná relé jsou použita ke spínání rezistorové sítě (Obr. 1.1), která je podstatou celého zařízení. Silový spínací obvod byl řešením mého předešlého semestrálního projektu (v příloze na DVD). Jednotka odporové zátěže s řadou rezistorů o hodnotách uvedených v tabulce 1.1, má dle výrobce nominální ztrátový výkon 2500 W a nominální napětí 200 V.

Obr. 1.1: Síť výkonových rezistorů

Tab. 1.1: Hodnoty impedancí výkonových rezistorů

$R_1 = 400\ \Omega$
$R_2 = 400\ \Omega$
$R_3 = 200\ \Omega$
$R_4 = 100\ \Omega$
$R_5 = 50\ \Omega$
$R_6 = 25\ \Omega$
$R_7 = 12{,}5\ \Omega$
$R_8 = 6{,}25\ \Omega$

Další podmínkou návrhu bylo použití programovatelného logického obvodu FPGA, tedy přesněji programovatelného hradlového pole Altera MAX EPM7128S. Zmíněný obvod by v sobě měl obstarávat logiku řízení a komunikace. Implementovaný řídicí program by měl být psán v jazyce VHDL. Ale hlavně by zařízení mělo být schopno komunikovat s PC. Uživatel bude mít tak možnost volit parametry elektrického odporu na výstupních svorkách zařízení, buď pomocí aplikace na počítači, nebo pomocí ovládacího panelu na zařízení samotném. Na základě poskytnutých parametrů a paralelního uspořádání jednotlivých rezistorů, by zařízení mělo být schopno měnit svou vnitřní impedanci v 256 hladinách popsaných v příloze (Příloha A).

2. Teoretický úvod

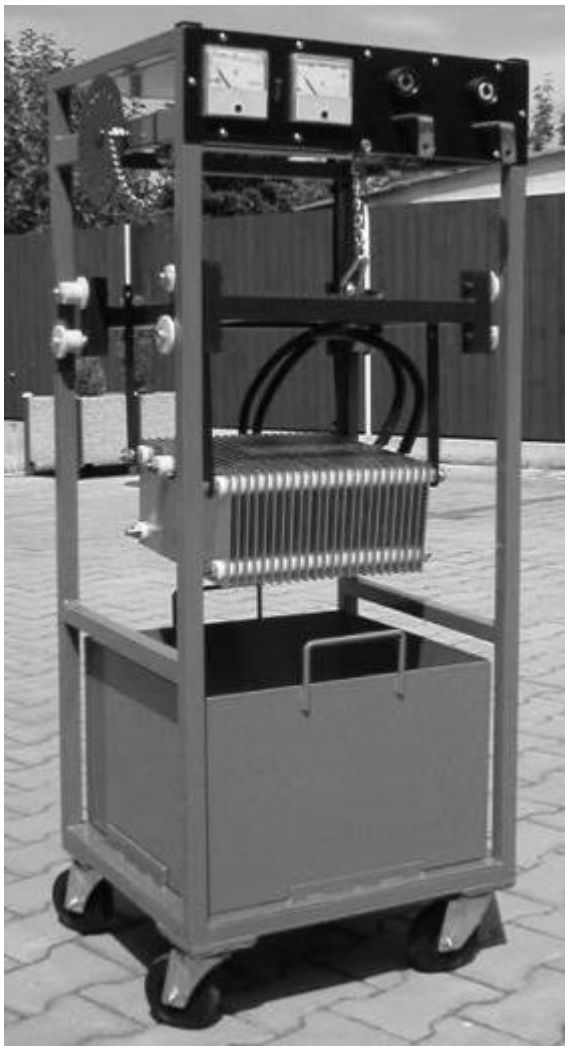
V této části jsou přiblíženy oblasti, které přímo souvisí s návrhem odporové zátěže. Jde především o programovatelná hradlová pole FPGA, komunikaci po univerzální sériové sběrnici USB, a USB převodník s technologií firmy FTDI. To jsou oblasti, na jejichž vlastnosti a možnosti bude odkazováno v dalších kapitolách této práce. A je tedy třeba provést stručné seznámení s těmito tématy.

2.1. Dostupné přepínané pasivní odporové zátěže

Všechny odporové zátěže můžeme z hlediska jejich konstrukce a funkce rozdělit na pasivní a aktivní. Pasivní zátěže jsou převážně konstruovány za pomoci výkonových drátových výkonových rezistorů. Soustavy těchto výkonových rezistorů pak bývají označovány jako rezistorové banky (z angl. Resistor banks) (Obr. 2.1). Zátěž je tvořena sadou výkonových odporových segmentů, připínaných převážně pomocí relé (popř. stykačů) do paralelní sítě. Nevýhodou pasivních zátěží je, že nelze plynule regulovat jejich odpor. Jedná se tedy o regulaci skokovou, pomocí různých odboček odporových smyček.

Obr. 2.1: Výkonová rezistorová banka

Další možností konstrukce pasivní odporové zátěže je tzv. vodní zátěž (Obr. 2.2). Zařízení založená na této konstrukci se používají jen zřídka. Výstupy této zátěže jsou připojeny na elektrody, které jsou pomocí mechanismu plynule ponořovány, nebo vynořovány z elektrolytu. Tím je zajištěna plynulá regulace odporu. Elektrolytem může být obyčejná voda s různými příměsemi. Pro dosažení požadovaného odporu a tím taktéž protékaného proudu je nutné použít co největší plochu elektrod. Toho lze například dosáhnout použitím několika deskových elektrod vedle sebe a paralelním propojením lichých a sudých desek.



Obr. 2.2: Provedení odporové zátěže firmy JK-WELD s.r.o.

Druhou skupinu zátěží tvoří tzv. aktivní zařízení. Tyto řízené zátěže bývají označovány také jako elektronické zátěže (Obr. 2.3). Dělí se podle provedení na stejnosměrná, střídavá a kombinovaná zařízení. Tyto aktivní zátěže mají výhodu oproti pasivním zejména v rozměrech a hmotnosti. Hmotnost a rozměry elektronických zátěží jsou mnohokrát menší. Ovšem jejich konstrukce je podstatně komplikovanější a jejich cena je samozřejmě mnohokrát vyšší. [10]

Obr. 2.3: Elektronická zátěž Chroma

Příklad parametrů elektronické zátěže firmy Chroma:

Chroma Programable AC/DC Electronic load 63800

- Výkony: 1800W, 3600W, 4500W
- Pracovní napětí: 50 V ÷ 400V
- Maximální pracovní proudy: 18 A, 36 A, 45 A
- Špičkové proudy: až 135 A
- Frekvence: 45 Hz ÷ 440 Hz, nebo DC

V příloze na DVD je k nahlédnutí celý prospekt tohoto zařízení.

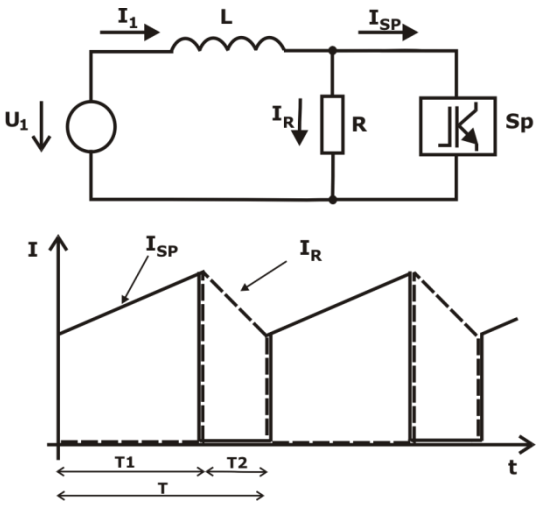
Toto zařízení se schopno kupříkladu simulovat nejen odporovou, ale i indukční nebo kapacitní zátěž.

2.1.1. Pulzní měnič odporu

Jedním z možných principů elektronické zátěže je pulzní měnič [6]. Schéma zapojení je na obrázku 2.4. Nezbytnou součástí obvodu je tlumivka, která omezí strmost nárůstu proudu při spínání. Je-li spínač Sp sepnutý (interval T_1), proud I_1 se uzavírá přes tento spínač a proto proud roste. Jakmile je spínač vypnut (interval T_2), proud I_1 se uzavírá přes odpor a proud klesá. V následujícím stavu prochází obvodem střední hodnota proudu I_1, která odpovídá ekvivalentnímu odporu Re. Platí:

$$U_1 \cdot I_1 \cdot T = R \cdot I_1^2 \cdot T_2 \quad \longrightarrow \quad I_1 = \frac{U_1}{R \cdot \frac{T_2}{T}} = \frac{U_1}{R_e}$$

$$R_e = R \cdot \frac{R_2}{T} \quad \longrightarrow \quad 0 \leq R_e \leq R$$



Obr. 2.4: Pulzní měnič odporu

Měnič je řízen prostřednictvím poměrné doby sepnutí měniče. Řízení může probíhat různými způsoby:

- Konstantním kmitočtem spínání
- Dvouhodnotové řízení
- Řízení s konstantní dobou sepnutí

Pokud odporové zátěže (jak pasivní tak aktivní) umožňují komunikaci s PC, tak je převážně využíváno sběrnic USB, GPIB, nebo RS232. Pomocí PC je možné měřit aktuální parametry zátěží, jako je napětí, protékaný proud a elektrický odpor. K řízení těchto zátěží se převážně používají speciální uživatelské aplikace, ovšem mohou komunikovat například i s různými výpočetními platformami, jako je např. Matlab.

Protože nejrozšířenější a nejpoužívanější komunikační sběrnicí je sériová sběrnice USB, tak i já jsem zvolil pro účely vyvíjeného zařízení toto komunikační rozhraní.

2.2. Komunikační rozhraní USB

Jak vyplývá z názvu této sběrnice, data jsou zde přenášena sériově po jednotlivých bitech a to diferenčně. Tento způsob přenosu do jisté míry snižuje rušení okolním prostředím. Datové vodiče nesou vzájemně negované signály o napěťových úrovních 0 ÷ 3,3 V.

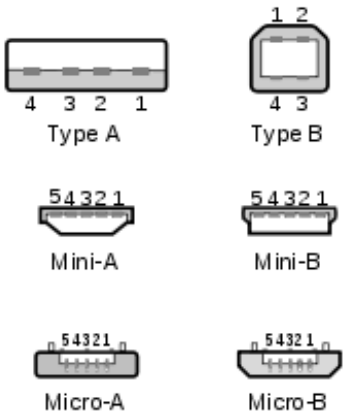
Nespornou výhodou je možnost připojování zařízení „plug&play“. To znamená automatickou detekci a konfiguraci zařízení v okamžiku připojení bez nutnosti restartování počítače nebo instalování ovladačů daného zařízení.

Ve verzi USB 1.1 v režimu Low-Speed (tedy v režimu pomalých zařízení), je možno dosáhnout přenosové rychlosti 1,5 Mbit/s, v režimu rychlých zařízení Full-Speed až 12 Mbit/s. Tato přenosová rychlost byla pro moderní periférní zařízení počítačů naprosto nedostačující. Proto sběrnice prošla v roce 2000 inovací (USB 2.0) a nabízí tak maximální přenosovou rychlost 480 Mbit/s (high speed). Zařízení ve verzi 2.0 jsou zpětně kompatibilní s verzí USB 1.1. V současné době se začíná rozšiřovat také třetí verze (USB 3.0), která byla úspěšně dokončena již v roce 2008. Nová verze protokolu USB 3.0 umožňuje více jak desetinásobné zrychlení komunikace. Její teoretická přenosová rychlost může dosahovat úrovně až 5 Gbit/s. Nová verze USB se také liší počtem vodičů. Je osmivodičová namísto původního provedení, které je čtyřvodičové. [12]

Rozlišení typu zařízení na full speed, nebo high speed, resp. low speed režim se zajišťuje hardwarově. Tedy připojením pull-up rezistoru 1,5 kΩ k lince D+ resp. D-.

2.2.1. Fyzická vrstva sítě

Konektory sběrnice USB mají různé provedení (*Obr. 2.5*).



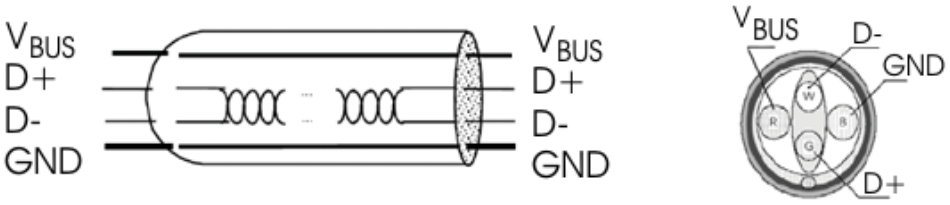
Obr. 2.5: USB konektory

Tab. 2.1: Zapojení USB konektoru

Pozice	Funkce
1	+5 V
2	Data (přímá)
3	Data (negovaná)
4	GND (zem)

Zmíněné parametry (Tab. 2.1) platí pro USB 1.1 a 2.0

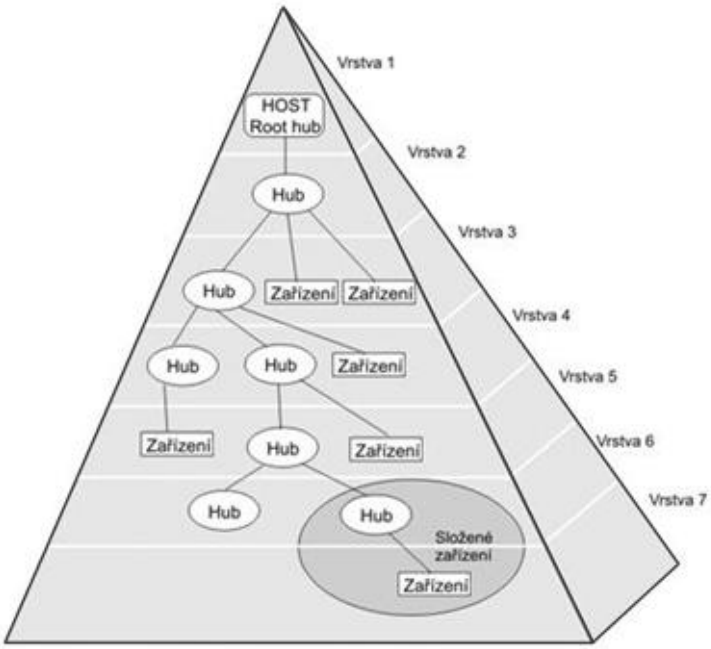
Pro připojování zařízení na sběrnici se doporučuje kabel o maximální délce 5 metrů, není-li využito tzv. opakovačů, čili obdoby zesilovače. Jak již bylo zmíněno, kabelový svazek vodičů kromě vodičů napájení a země obsahuje ještě dva datové vodiče. Tyto datové vodiče vytvářejí jeden pár diferenčního vodiče. Svazek datových vodičů může být kroucený a stíněný, což opět redukuje elektromagnetické rušení z okolí USB kabelu (Obr. 2.6). [8],[9],[12]



Obr. 2.6: Průřez USB kabelem

2.2.2. Topologie sítě

Sběrnice USB má hvězdicovou strukturu (Obr. 2.7). Celá síť je tvořena třemi základními elementy: USB host, USB hub a funkční jednotkou, tedy samotným zařízením. Tyto elementy jsou přímo propojeny (point to point). Struktura umožňuje připojit až 127 zařízení (funkčních jednotek) k jednomu hostiteli. Ve verzi USB 1.1 do páté úrovně vrstvy a ve verzi 2.0 až do úrovně sedmé. [7]



Obr. 2.7: Topologie USB sběrnice

Huby (rozbočovače) mají z hlediska uživatele důležitou funkci, je jí připojení více zařízení zároveň. Huby poskytují podřízeným zařízením více přípojných portů z jednoho portu nadřazeného systému. Hub také zásobuje zařízení napájecím napětím. Při startu může každé USB zařízení odebrat maximálně 100 mA. Pokud toto nestačí, žádá zařízení o zvýšení proudu pomocí konfiguračního deskriptoru, celkově lze odebrat proud maximálně až 500 mA. Externí hub může pro zařízení napájené ze sběrnice USB dodávat proud pouze 100 mA, protože celkový odběr nesmí přesáhnout 500 mA a samotný hub má také určitou spotřebu. [4]

Funkční jednotka je USB zařízení, schopné vysílat a přijímat datové nebo řídící pakety v rámci sběrnice. Typickou implementací funkce je samostatné zařízení, které se připojuje do systému kabelem. Ovšem ne každé zařízení připojitelné k USB je pouhá funkční jednotka. Tento celek může tvořit i více prvků v jednom zařízení s integrovaným USB hubem. Takové zařízení se hostitelskému řadiči jeví jako hub s určitým počtem neodpojitelných zařízení. [7]

Firmware každé funkční jednotky obsahuje konfigurační informace, které popisují možnosti, vlastnosti a požadavky zařízení. Tyto informace se využívají při tzv. enumeraci periferie při připojení ke sběrnici, čili rozpoznání zařízení. Jedná se například

o požadavky na určitou šířku pásma nebo o alokování zdrojů potřebných ke správné funkci zařízení.

2.2.3. Načítání parametrů a rozpoznání zařízení

Po připojení zařízení se operační systém dotáže na jeho parametry. Zařízení musí být automaticky rozpoznáno operačním systémem. To je nutné pro výběr odpovídajících ovladačů, mimo prvotního připojení zařízení, při kterém jsou ovladače do systému nainstalovány. Poté je zařízení ohlášeno a získá svou sběrnicovou adresu.

Při enumeraci, tedy načítání parametrů zařízení, se operační systém dotáže nově připojeného zařízení na určité parametry, tzv. deskriptory - přesně definované bloky dat. (Tab. 2.2). Hub rozpozná připojení nového zařízení zdvihnutím napěťové úrovně linky D+ nebo D-. Následují tyto kroky:

- Hub informuje hostitelský počítač, že je připojeno nové zařízení.
- Počítač se dotáže hubu, na který port je zařízení připojeno.
- Počítač nyní aktivuje tento port a resetuje sběrnici USB.
- Hub vyvolá USB reset o délce 10 ms a uvolní pro zařízení proud 100 mA. Jednotka Seriál Interface Engine následně vyvolá reset mikrokontroléru a zařízení je připraveno k použití.
- Mezitím počítač přiřadí zařízení specifickou síťovou adresu a načte všechny informace obsažené v deskriptoru zařízení.

Tab. 2.2: Nejdůležitější vlastnosti zařízení v jeho deskriptoru

Položka	**Význam**
VID (Vendor ID)	16bitový číselný identifikátor výrobce (číslo přidělované organizací USB)
PID (Product ID)	16bitové číslo určené výrobcem
Manufactured ID	Řetězec identifikující výrobce
Manufacturer	Řetězec popisující výrobce
Product	Řetězec popisující výrobek
Serial Number	Sériové číslo (umožní připojení několika stejných výrobků)
Charakter napájení	Napájení ze sběrnice + odebíraný proud, vlastní napájení

Shrnutí základních parametrů:

- Sériové rozhraní
- Přenosové rychlosti: 1,5 Mb/s; 12 Mb/s, 480 Mb/s a 5 Gbit/s.
- Možnost připojování zařízení na relativně velkou vzdálenost (až 5 metrů)
- Možnost napájení zařízení přímo ze sběrnice (běžně lze odebrat 100 mA, po speciálním přihlášení až 500 mA)
- Velký počet připojitelných zařízení (při použití hubu až 127)
- Podpora plug&play (připojování a odpojování zařízení za provozu)
- Podpora operačními systémy Windows XP/Vista/7 (také Linux, MAC OS-8, OS-9,OS-X)

2.2.4. Obvody pro USB

Existuje několik způsobů jak implementovat USB rozhraní do dané aplikace:

- Implementovat celý protokol do firmware v procesoru
- Využít procesor vybavený s USB rozhraním
- Využít speciálních jednoúčelových obvodů
- Využít univerzálních USB řadičů

Softwarová implementace do procesoru je řešením vývojově nejsložitějším. A obvykle toto řešení dosahuje menších přenosových rychlostí než ostatní možnosti implementace USB rozhraní. Procesor s integrovaným USB rozhraním sice nabízí vysoký komfort obsluhy komunikace, ale dostupnost těchto procesorů je problematická. Speciální jednoúčelové obvody nalézají využití při konstrukci zařízení, jimiž jsou např.: MP3 přehrávače, převodníky z USB na IrDA, některé tiskárny, a další počítačové periferie.

2.3. FTDI převodníky

Pro implementaci USB jsem se po dlouhé úvaze rozhodl pro převodník FT245BL firmy Future Technology Devices International Ltd. (FTDI).

Firma FTDI má ve své nabídce celou řadu USB převodníků. Nejzákladnějšími jsou obvody FT232 a FT245 s různými modifikacemi. Čímž se rozumí například více USB převodníků v jednom čipu, nebo jejich různá velikost a provedení pouzder. Pro implementaci těchto obvodů je potřebný minimální počet externích součástek, což opět zjednodušuje implementaci obvodu.[2], [3], [16]

Základní integrované obvody řady FT:

- FT232 (převodník UART ↔ USB)
- FT245 (převodník 8bit FIFO brána ↔ USB)

Integrovanými obvody firmy FTDI lze pomocí USB snadno připojit široké spektrum aplikací. Základní USB protokol se před vývojářem skrývá. Z hlediska PC se jeví jako standardní sériový port, ale z hlediska zařízení buď jako standardní RS232 (FT232), nebo obousměrná osmibitová sběrnice (FT245).

2.3.1. Paralelní USB převodník FT245

Obvod svými parametry plně vyhovuje požadavkům mé aplikace. Tedy požadavkům na přenosovou rychlost, připojení k programovatelnému logickému obvodu a jednoduchost řízení komunikace. Výrobce FTDI poskytuje navíc ke všem svým produktům rozsáhlou podporu. Poskytuje rozsáhlou dokumentaci k vyráběným obvodům a příklady zdrojových kódů pro obslužné aplikace v několika nejrozšířenějších programovacích jazycích (C+, C#, Delphi, LabView, Matlab, Java, Python…). Samozřejmě jsou k dispozici také ovladače pro všechny systémy MS Windows i Linux. Jako softwarový nástroj přístupu k obvodům slouží DLL knihovna funkcí, která je rovněž volně šiřitelná.

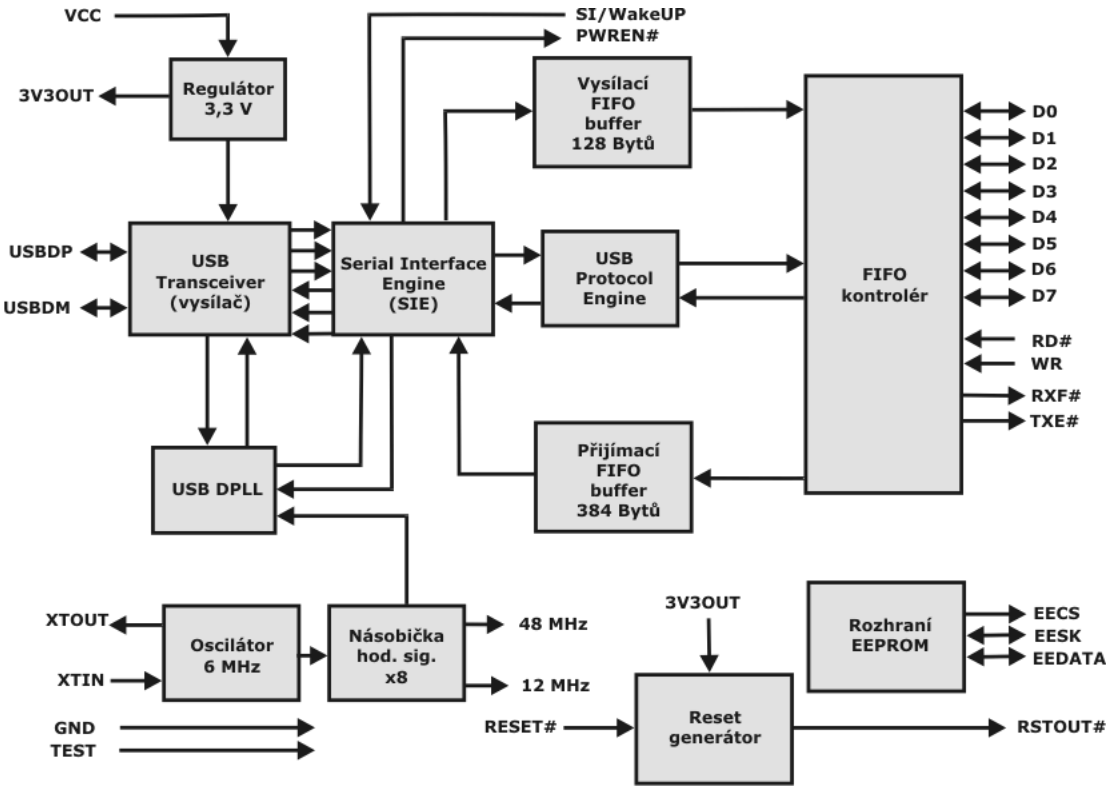
Tento integrovaný obvod umožňuje obousměrný přenos mezi USB sběrnicí a osmibitovým výstupním datovým portem obvodu FT.

Obr. 2.8: USB převodník FT245BL

2.3.2. Vlastnosti integrovaného obvodu FT245BL

Blokové schéma obvodu

Toto blokové schéma je kompletně popsáno v datasheetu obvodu FT245BL (na přiložené DVD). Zde ho uvádím pouze pro představu vstupních a výstupních signálů. [16]



Obr. 2.9: Blokové schéma obvodu FT245BL

Nejdůležitější vstupy a výstupy obvodu:

- **USBDP,USBDM** - datové linky sběrnice USB
- **XTOUT, XTIN** – vstup a výstup externího zdroje hodinového signálu
- **D0 ÷ D7** – datový komunikační port
- **RD#,WR,RXF#,TXE#** - řídicí signály komunikace
- **EECS, EESK, EEDATA** – vstupy a výstupy pro externí EEPROM

Obvod FT245BL je kompatibilní s USB 1.1 i USB 2.0. Podporuje přenosovou rychlost až 12 Mbit/s. Jak bylo již výše řečeno (kapitola 2.3), detaily komunikace a USB protokol jsou před uživatelem do jisté míry skryty. Navenek se jeví jako osmibitový port s funkcí FIFO registru. Ostatní vstupy a výstupy obvodu řídí

komunikaci obvodu s USB. K čipu je možné připojit externí paměť EEPROM, jejíž obsah určuje parametry obvodu potřebné pro jeho enumeraci (viz kapitola 2.2.3). Pro FT obvody je pouze dáno licencované číslo VID (0403), ostatní parametry jsou libovolně volitelné.

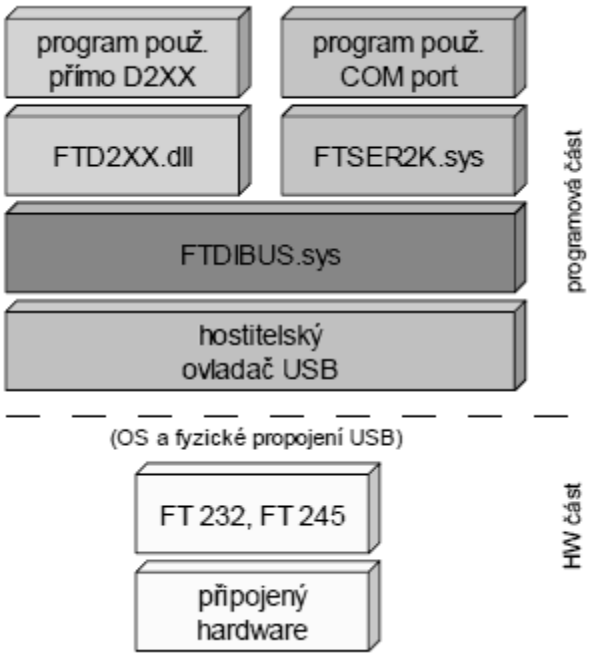
Není-li použita externí paměť, pracuje obvod v základním nastavení. Její použití se však obecně doporučuje. V případě připojení více zařízení s FT převodníky k jednomu PC, nelze vyloučit bezchybné rozlišení těchto zařízení. Není možné rozeznat jejich sériová čísla.

Všechny potřebné parametry je možné do paměti EEPROM nahrát za pomoci utilit poskytovaných FTDI (např. aplikací MProg).

2.3.3. USB ovladače FT převodníků

Firma FTDI nabízí pro komunikaci FT převodníků s PC dva druhy ovladačů (Obr. 2.10). První možností je využití ovladačů Virtual Com Port (VCP), jenž umožňují přistupovat k obvodu FT jako k imaginárnímu portu počítače. Tento typ ovladačů je vhodné použít pro programování pomocí API funkcí.

Druhou z možností je instalace ovladačů D2XX (USB Direct Drivers + DLL software interface), ty umožňují přistupovat k obvodu FT takzvaně přímo. To podstatně rozšiřuje paletu nabízených funkcí obvodu. Například tyto ovladače umožňují měnit obsah externí EEPROM paměti převodníku. Tyto ovladače pak využívají distribuovanou knihovnu funkcí FTD2xx.dll.

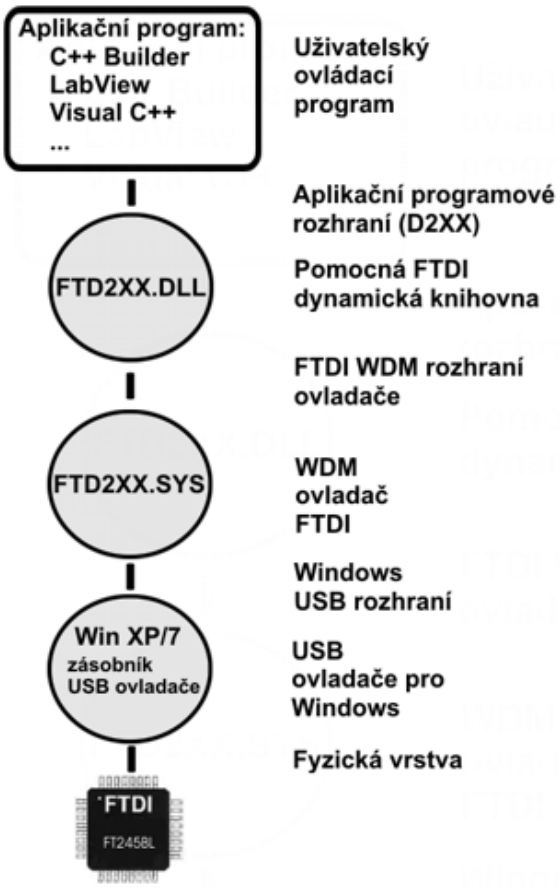


Obr. 2.10: Blokové schéma dvojího přístupu k FT převodníku

Výrobci různých zařízení s FT převodníky často vytvářejí vlastní dll knihovny. Jak jsem ale zjistil, jedná se převážně o redukovanou knihovnu FTD2xx.

Knihovna FTD2xx obsahuje WDM ovladač FTD2xx.sys (Obr. 2.11), který komunikuje se zařízením přes Windows USB zásobník. Knihovna FTD2xx je rozhraním pro aplikační programy vytvářené ve vývojových prostředích Visual C++, LabView, Delphi, Visual Basic, Matlab…atd.

Na přiloženém DVD jsou instalační soubory zmiňovaných ovladačů a program pro jejich deinstalaci.



Obr. 2.11: Blokové schéma přímého přístupu k převodníku

V příloze této práce (Příloha A) je také podrobný návod k instalaci ovladačů.

2.3.4. Funkce dynamické knihovny

Základní funkce zajišťují přístup k zařízení vybavenému obvodem FT. Před tím, než se začne se zařízením pracovat, musíme získat jeho handle. Otevření zařízení (získání jeho handle) zajišťují funkce FT_Open a FT_OpenEx. Poté je možno vysílat/číst data funkcemi FT_Write a WT_Read. Po dokončení operací je nutné zavřít zařízení voláním funkce FT_Close.

Kromě základních funkcí jsou k dispozici další funkce: FT_ResetDevice (provede USB reset zařízení), FT_Purge (vyprázdní přijímací nebo vysílací buffer), FT_SetTimeouts (nastaví komunikační time-outy), FT_GetQueueStatus (zjistí stav přijímací fronty), FT_GetStatus (zjistí stav zařízení). [2],[3]

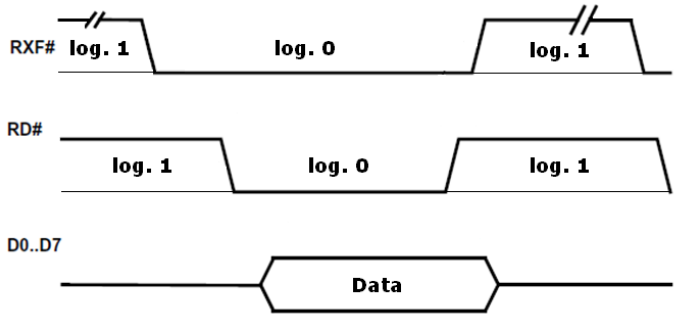
Dále je možno získat výčet všech zařízení připojených k počítači pomocí funkce FT_ListDevices. Lze nastavovat charakteristiky přenosu dat, hw/sw handshake, řídicí signály modemu pomocí funkcí: FT_SetBaudRate - nastavení komunikační rychlosti (nestandardní přenosové rychlosti pak pomocí funkce FT_SetDivisor), FT_SetDataCharacteristics, … a mnoho dalších funkcí.

Všechny dostupné funkce jsou obsaženy v programátorské příručce. (na přiloženém DVD)

2.3.5. Režim BitBang

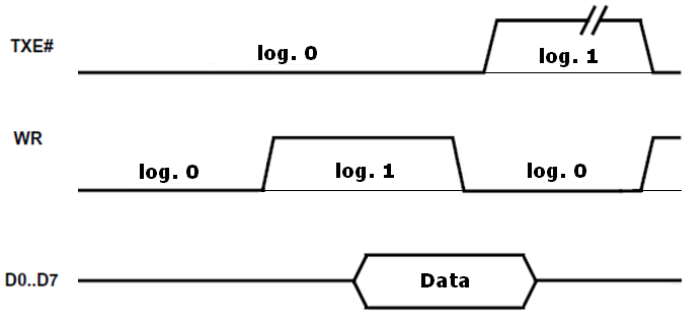
Standardně se řídí přenos dat mezi převodníkem a USB sběrnicí řídicími vstupy a výstupy obvodu. (RXF, RD, TXE,WR)

Časová posloupnost řídicích signálů pro čtení z datového bufferu obvodu.



Obr. 2.12: Časový diagram pro čtení z datového portu FT převodníku

Časová posloupnost řídicích signálů pro zápis do datového bufferu obvodu.



Obr. 2.13: Časový diagram pro čtení z datového portu FT převodníku

Tyto datové přenosy jsou podrobně popsány v datasheetu obvodu FT245BL (na přiloženém DVD).

Některé FT obvody podporují zvláštní režim přenosu dat. Ten řídicí signály ke své funkci nepotřebuje. Jednotlivé bity datového bufferu jsou pak pevně nastaveny jako vstupní nebo výstupní, dle libosti. Tento režim se nazývá BitBang mód. Komunikace převodníku FT s řídicí logikou zařízení se tak podstatně zjednodušuje a pro jednoduché aplikace plně postačuje.

V režimu BitBang se datové pakety posílají na vstupně/výstupní buffer sekvenčně a to rychlostí danou nastavením předděličky přenosové rychlosti. Obvod lze použít v jednoduchých aplikacích jako vstupně/výstupní řadič. Například pro ovládání světel, relé a stykačů.

Přepínání BitBang módu a klasického řízení toku dat, umožňuje vytvářet generické USB periferie. To znamená periferie, jejichž funkce se mění aplikačním programem. Například po připojení FPGA je po resetu tento obvod bez funkce, aplikační program může v režimu BitBang provést nahrání konfiguračních dat do FPGA a tím definovat jeho funkci „naprogramovat ho“. Poté se lze přepnout do normálního režimu a naprogramovaný FPGA pak může klasicky komunikovat s počítačem přes USB. [12]

2.4. Programovatelná hradlová pole

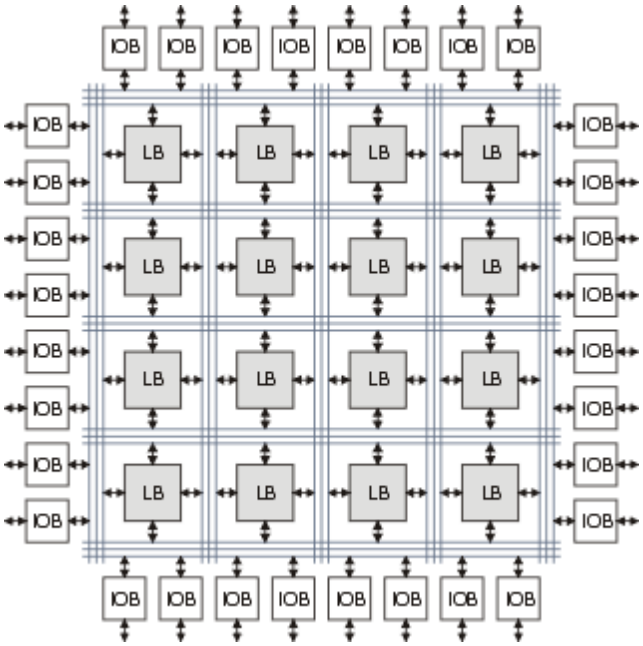
Čili obvody FPGA (Field Programmable Gate Array) patří do skupiny programovatelných logických obvodů. Všechny číslicové programovatelné obvody se souhrnně označují jako PLD (Programmable Logic Device).

Programovatelné logické obvody jsou založeny na jednoduchém principu, jehož základní myšlenkou je ten fakt, že jakoukoliv logickou funkci lze nahradit odpovídajícím počtem základních logických hradel. K těmto účelům se nejvíce používá dvouvstupové logické hradlo NAND. Současné největší obvody FPGA obsahují až 6 milionů ekvivalentních hradel.

Základním prvkem FPGA obvodů jsou malé logické buňky, obvykle se čtyřmi až šesti vstupy, z nichž jsou logické funkce vytvářeny strukturou zvanou LUT (Look Up Table). Tyto malé logické buňky připomínají paměť PROM. Pro vytváření složitějších funkcí, se musí tyto buňky propojovat. Propojovací struktura zaujímá významnou

plochu čipu. Logické buňky se sdružují do složitějších bloků. Tyto bloky bývají označovány jako CLB (Configurable Logic Block), nebo LB (Logic Block).

Typickou blokovou strukturu FPGA znázorňuje následující obrázek 2.14.



Obr. 2.14: Bloková struktura FPGA obvodu

Bloky s názvem IOB jsou bloky vstupně/výstupních obvodů každého pinu FPGA obvodu. IO bloky obvykle obsahují budič, registr, multiplexer a ochranné obvody. Tyto bloky obsluhují vstupní a výstupní piny FPGA obvodu, zajišťují dodržení logických úrovní používaných vně obvodu a potřebnou zatížitelnost. Kromě bloků znázorněných na předchozím obrázku, integrují výrobci do FPGA další prvky. Těmi jsou například:

- podpůrné struktury pro aritmetické operace
- bloky pro úpravu hodinových signálů
- procesorová jádra
- struktury pro podporu různých vstupně/výstupních standardů
- bloky pro podporu rychlé sériové komunikace
- bloky pro řízení odběru
- a řada dalších pomocných bloků a funkčních celků

FPGA obvykle umožňují propojit signály logických bloků přímo se sousedními, bez nutnosti využívat globální propojovací matici. Toto spojení má mnohem menší

zpoždění a umožňuje realizovat například rychlé obvody šíření přenosu, což je nezbytné pro sčítačky nebo násobičky.

2.4.1. Programování obvodů FPGA

Pro práci s programovatelnou logikou je nutné používat počítačový návrhový systém. Tento systém automaticky zajistí celou řadu různých úkonů, počínaje editací vstupních souborů, představujících popis vyvíjené aplikace, přes syntézu odpovídajícího elektrického zapojení a implementaci do cílového obvodu. Umožňuje taktéž simulaci aplikace. [5]

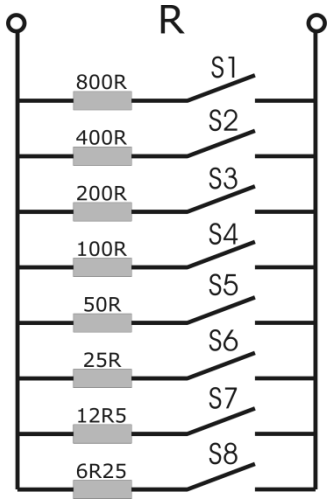
Vstupní podklady, sloužící k programování obvodů FPGA, mají nejčastěji textový nebo grafický charakter. Textový způsob programování v současné době nejčastěji používá některý z jazyků HDL (Hardware Design Language). Nejpoužívanější bývají jazyky VHDL a Verilog. Výhodou textového stylu programování je možnost využití jednoduchých textových editorů pro vytváření a prohlížení vyvíjených programů. Výhodou je také jejich přenositelnot mezi různými návrhovými systémy. Jazyky HDL umožňují zapisovat program v tzv. behaviorálním stylu, který srozumitelně (pro programátora) popisuje chování obvodu.

Ke grafickým prostředkům programování patří především schémata obvodu. Nevýhodou tohoto popisu schématem je jeho nepřenositelnost mezi návrhovými systémy. Návrhové systémy běžně umožňují vygenerování odpovídajícího textového popisu k danému schématu obvodu. [13]

3. Realizace zařízení

3.1. Výkonová část obvodu

Jak již bylo řečeno, tato část obvodu je dílem mého předcházejícího semestrálního projektu [15], a proto se pokusím pouze o stručný popis tohoto obvodu, jehož hlavním úkolem je připojování nebo odpojování osmi výkonových rezistorů z paralelní sítě pomocí relé. Na obrázku vedle (Obr. 3.1) je tato síť výkonových rezistorů naznačena. Zobrazené spínače jsou



Obr. 3.1: Zjednodušené schéma výkonové části

ekvivalentem spínacích kontaktů relé, které tyto větve rezistorů obsluhují. Byla zde použita relé pro střídavé proudy, avšak obvod umožňuje, díky navrženému odlehčovacímu (zhášecímu) obvodu, spínání a rozpínání i stejnosměrných proudů.

3.1.1. Redukce jiskření

Při rozpínání kontaktů relé, jimiž prochází proud, dochází k ionizaci okolního prostředí silových kontaktů a dochází ke vzniku elektrického oblouku. V okamžiku rozpínání se kontaktní tlak zmenšuje, zmenšují se styčné plochy, až se styk kontaktů přeruší úplně. Těsně před tímto okamžikem rozpojení, dochází ke zvýšení teploty na styčných plochách až do té míry, že se kov tvořící kontakty odpařuje. Energie nahromaděná
v indukčnostech a kapacitách rozpínaného obvodu pak způsobuje přepětí. Se zvětšující se vzdáleností kontaktů, způsobuje toto přepětí elektrický oblouk. Ten samozřejmě způsobuje přenos materiálu mezi kontakty a samotné kontakty tím nenávratně ničí.

Vypínání stejnosměrných proudů je z principu složitější, než vypínání střídavých proudů. U střídavých proudů se ve své podstatě oblouk po rozpojení kontaktů relé zháší sám, protože v jedné periodě prochází okamžitý proud hned dvakrát nulovou hodnotou, tj. při frekvenci 50Hz projde nulou stokrát za sekundu. Tím pádem se oblouk sám přeruší, nicméně vzniká zde také. Tato výhoda bohužel samozřejmě neplatí pro stejnosměrné proudy.

Aby se kontakty relé nadměrně neopotřebily, je nutné dobu hoření oblouku zkrátit umělým zhášením.

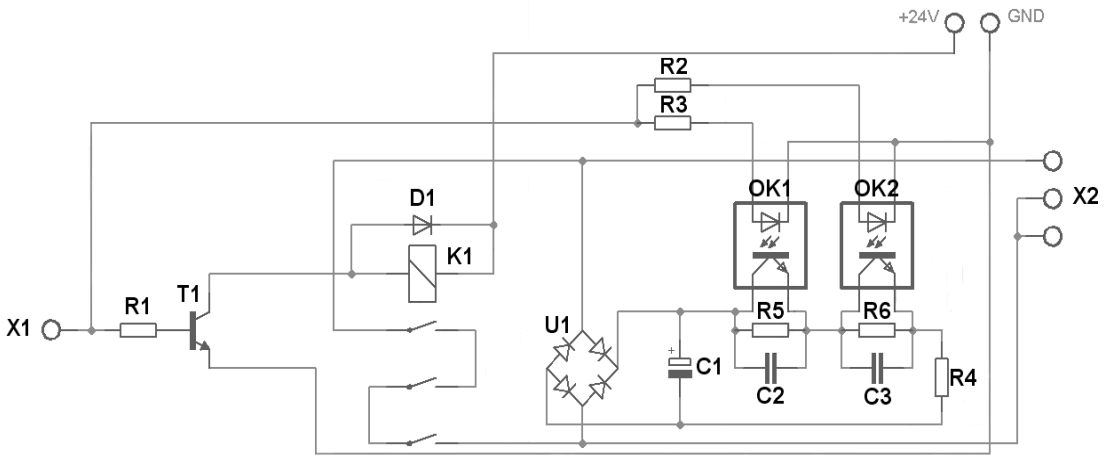
Způsoby zhášení elektrického oblouku:

- rychlým oddálením kontaktů
- vyfouknutím oblouku magnetickým polem do zhášecí komory
- v olejové lázni a různých jiných prostředích
- rychlým snížením přepětí
- rozdělením oblouku (můstkové kontakty)

Právě poslední dva způsoby eliminace oblouku využívá zmiňovaný silový obvod.

3.1.2. Popis schématu výkonové části obvodu

Níže je popsán segment výkonové části obvodu (Obr. 3.2), který spíná jednu smyčku paralelní sítě rezistorů (viz kapitola 1). Celé schéma výkonové části obvodu je v příloze.



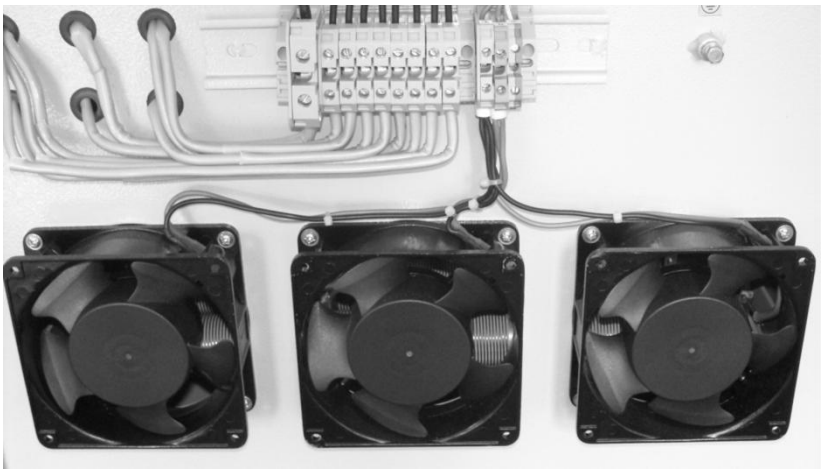
Obr. 3.2: Segment silové části obvodu, obsluhující jedno relé

Na svorku X1 (Obr. 3.2) se přivádí signál z řídicí části obvodu, tranzistor T1 spíná cívku relé K1 a jako ochrana tranzistoru před přepětím, které vznikne vždy při rozpínání relé, je k cívce antiparalelně zapojena usměrňovací dioda D1, která toto přepětí na tranzistoru eliminuje.

Zbytek tohoto obvodu (tzv. snubber) zajišťuje redukci jiskření, resp. vznik elektrického oblouku mezi kontakty relé. Kontakty relé jsou spojeny do série, protože se tím oblouk, který by vznikl při rozpínání na jednom kontaktu, „rozdělí“ na tři části a dochází k jeho „natažení“, to znamená rychlejšímu ochlazení a zhášení. Dále je na tyto sériově spojené kontakty paralelně připojen můstkový usměrňovač U1, který vzniklé přepětí při rozpínání usměrní a nabije se tak kondenzátor C1, čímž dojde k rychlému snížení přepětí na kontaktech relé. Dále se musí při každém sepnutí relé sepnout i optočleny OK1 a OK2, které způsobují vybíjení kondenzátoru C1 přes odpor R4, takže kondenzátor je připraven k nabíjení (pojmutí energie) při dalším rozepnutí kontaktů relé. Tímto způsobem je elektrický oblouk redukován.

Na svorku X2 je připojen samotný výkonový rezistor připínaný do sítě ostatních rezistorů.

Relé a vůbec celá konstrukce výkonové části obvodu je projektována maximálně na 16 A střídavého proudu a napětí až 250 V. Ztrátový výkon, vzniklý na rezistorové síti, může tedy dosahovat řádů jednotek kW. Proto se výkonové rezistory chladí ventilátory (Obr. 3.3), které jsou spuštěny bezprostředně po sepnutí síťového tlačítka na ovládacím panelu zařízení.



Obr. 3.3: Chlazení výkonových rezistorů

3.2. Napájení zařízení

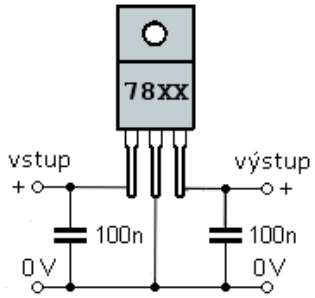
Stejně tak, jako silová část obvodu, byl napájecí obvod navržen a realizován v mém předešlém semestrálním projektu. Protože při návrhu napájecího obvodu nebylo ještě zcela jasné, jakou hodnotu napájecího napětí bude vyžadovat řídicí jednotka, tedy 3,3 V nebo 5 V, byl napájecí obvod vybaven regulovatelným zdrojem napětí. Obvod je realizován dle jednoduchého schématu, které je v příloze (Příloha H). A poskytuje stabilizované zdroje napětí +3 ÷ 8 V, +12 V a +24 V.

Napájecí obvod je vybaven aktivním chlazením v podobě ventilátoru, aby v případě vysokých ztrátových výkonů na zařízení, nedocházelo k výkyvům napájecích napětí.

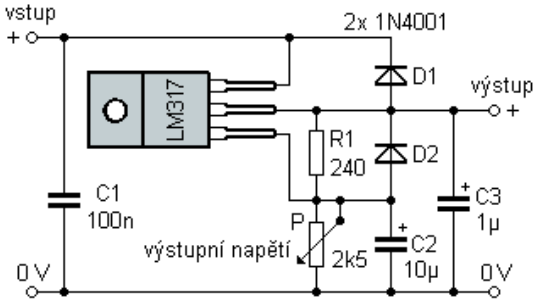
3.2.1. Popis schématu napájecího obvodu

Síťové napětí 230 V je sníženo transformátorem na napětí o efektivní hodnotě 24 V. Je usměrněno dvoucestným můstkovým usměrňovačem. A protože jsou v zařízení potřeba tři úrovně napětí (+5 V, +12 V a +24 V), je usměrněné napětí stabilizováno ve třech různých částech obvodu zdroje.

V první části je stabilizován zdroj napětí +12 V, který slouží k napájení chladících ventilátorů kompenzačních rezistorů a napájecího obvodu (Příloha D). Obsahuje integrovaný stabilizátor 7812 a je zapojen v základním zapojení doporučeném výrobcem (Obr. 3.4).



Obr. 3.4: Doporučené zapojení Stabilizátoru 78xx



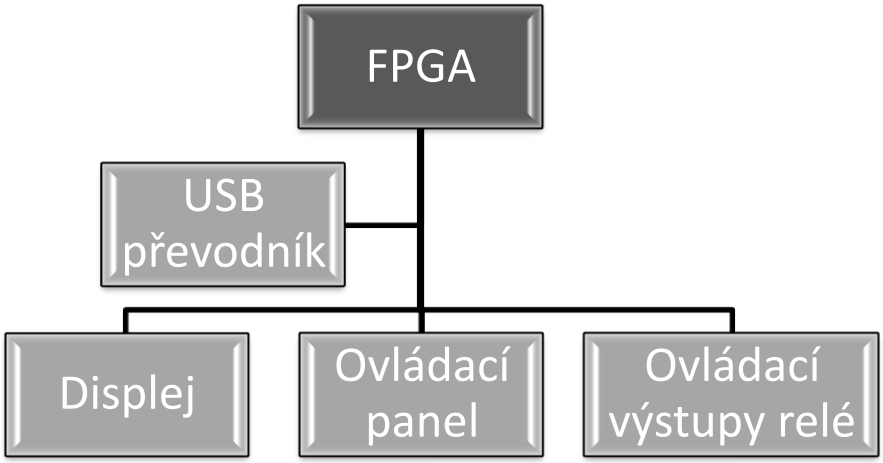
Obr. 3.5: Regulovatelný zdroj napětí s LM317

V další části obvodu je stabilizováno napětí +24 V. Napětí z můstkového usměrňovače je vyfiltrováno kondenzátorem 2200 uF a stabilizováno integrovaným stabilizátorem napětí 7824, který je schopen dodat maximální proud až 1,5 A. Což pro napájení cívek relé postačující (8 × maximálně 125 mA).

V poslední části zdroje je stabilizováno napětí +5 V pro napájení řídicí jednotky zařízení. Napětí za usměrňovacím můstkem je taktéž vyfiltrováno a je stabilizováno pomocí integrovaného obvodu LM317 (Obr. 3.5), díky tomuto obvodu je možno dodat do řídicí části obvodu maximální proud až 2 A s možností přesného nastavení výstupního napětí. Stabilita použitého obvodu je relativně větší než, kdyby byl použit obdobný integrovaný obvod jako v předchozích dvou případech (7805).

3.3. Řídicí jednotka zařízení

Nyní se konečně dostávám k řídicímu obvodu odporové zátěže. Tato část obvodu má za úkol řídit veškerou činnost celého zařízení. Na obrázku (Obr. 3.6) je znázorněno blokové schéma popisující řídicí obvod.



Obr.: 3.6: Blokové schéma řídicí jednotky

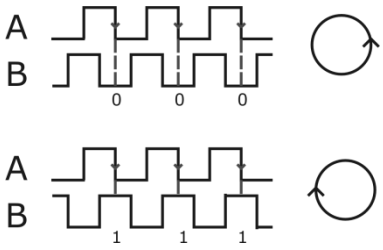
Jak již bylo řečeno v úvodu, pro implementaci řídicí logiky byl použit FPGA obvod Altera MAX EPM7128. Řídicí program pro tento obvod, který je psán v jazyce VHDL, je obsažen v příloze (Příloha E). Program obsluhuje všechny vstupní a výstupní signály. Všechny tyto signály jsou unifikovány v úrovních TTL, čili 0 V (log. 0) a +5 V (log. 1).

Obr. 3.7: Řídicí programovatelný logický obvod

Mezi vstupní signály patří dvoukanálový signál z enkodéru (Obr. 3.8), lze jím měnit hodnotu celkového odporu zařízení na výstupních svorkách a hodnotu zobrazenou na sedmisegmentovém LED displeji.

Pro přesnost připomenu, co je enkodér a jeho funkci. Enkodér je rotační součástka (čidlo), která generuje dva obdélníkové průběhy, ty jsou fázově posunuty přibližně o 90°. Pro vyhodnocení směru rotace enkodéru čteme při sestupné hraně prvního signálu (Obr. 3.9) stav druhého signálu a podle toho, zda je v log. 1 nebo log. 0, rozeznáváme směr rotace čidla.

Obr. 3.8: Rotační enkodér



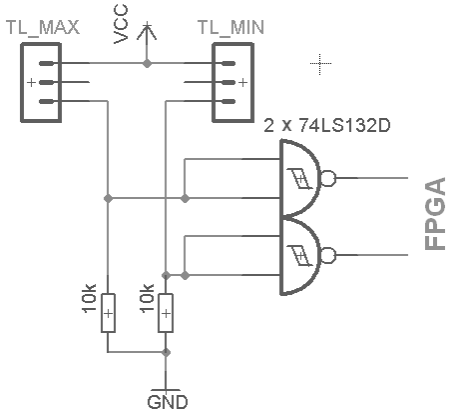
Obr. 3.9: Rozlišení směru rotace enkodéru

Zařízení je dále opatřeno dvěma tlačítky, jimiž lze nastavit mezní hodnoty maximum a minimum odporu. Hodnoty celkového odporu rezistorů nejsou zobrazovány na displeji přímo, ale slouží k tomu převodní hodnoty 0 ÷ 255. V převodní tabulce (Příloha A) připevněné na vrchním krytu zařízení jsou uvedeny hodnoty celkového odporu, odpovídající nastavené hodnotě na LED displeji. Osm výstupních signálů obvodu řídí spínání relé ve výkonové části obvodu. Poslední neméně důležitou periferií FPGA obvodu je USB převodník FT245BL, ten umožňuje komunikaci zařízení přes sériovou sběrnici USB s ovládací aplikací na PC.

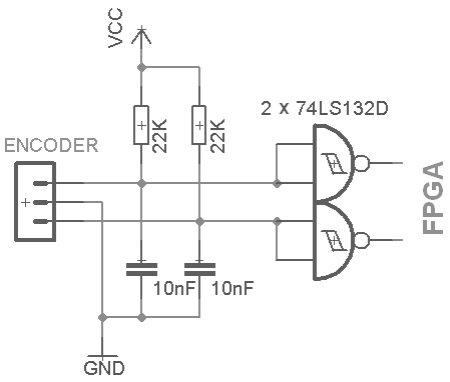
3.3.1. Popis schématu řídící jednotky

Schéma řídicí části obvodu viz Příloha F.

Programovatelné hradlové pole Altera MAX EPM7128 je řízeno hodinovým signálem z integrovaného oscilátoru frekvencí o velikosti 10 MHz. FPGA je napájeno napětím +5 V, nicméně tento obvod je schopen pracovat i s menším napájecím napětím (3,3 V).



Obr. 3.10: Tlačítka ošetřená proti zákmitu Schmittovým klopným obvodem

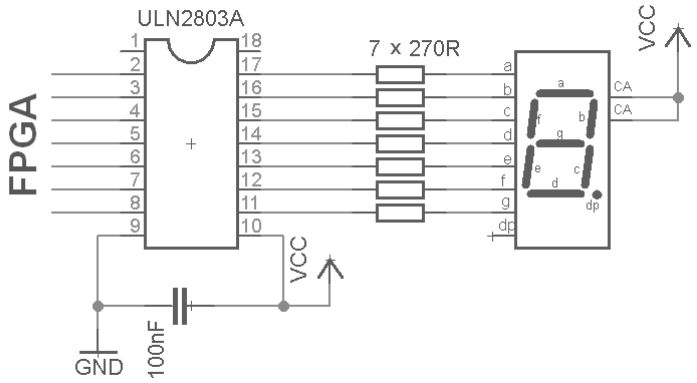


Obr. 3.11: Enkodér ošetřený proti zákmitu Schmittovým klopným obvodem

Jak bylo zmíněno výše, obvod obsahuje dvě tlačítka. Tato tlačítka jsou ošetřena proti zákmitu Schmittovým klopným obvodem (Obr. 3.10). Stejně jako tlačítka, jsou ošetřeny proti zákmitu i oba signály z enkodéru (Obr. 3.11).

Schmittovy klopné obvody spínají při náběžné hraně 1,7 V a při sestupné hraně 0,8 V. Obvodům nevadí pomalu rostoucí vstupní napětí a hodí se například na úpravu hran signálu, na převod pomalu rostoucího, či zašuměného signálu, využívají se v oscilátorech atd. Ošetření zákmitu tlačítek a enkodéru by samozřejmě bylo možné realizovat i softwarově v řídicí logice, tento způsob je ale poněkud rychlejší.

Výstupní signály FPGA, určené pro zobrazování výstupních hodnot nastavené úrovně odporu na sedmisegmentovém LED displeji, jsou zesíleny Darlingtonovými poli ULN2803 (Obr. 3.12). Tím je redukován odběr proudu z FPGA. Napájecí proud jednoho segmentu displeje je cca. 25 mA. Použitím zesilovacích prvků, je proud odebíraný z řídicího obvodu snížen na 2 mA. Taktéž je zesílen signál pro piezo měnič.



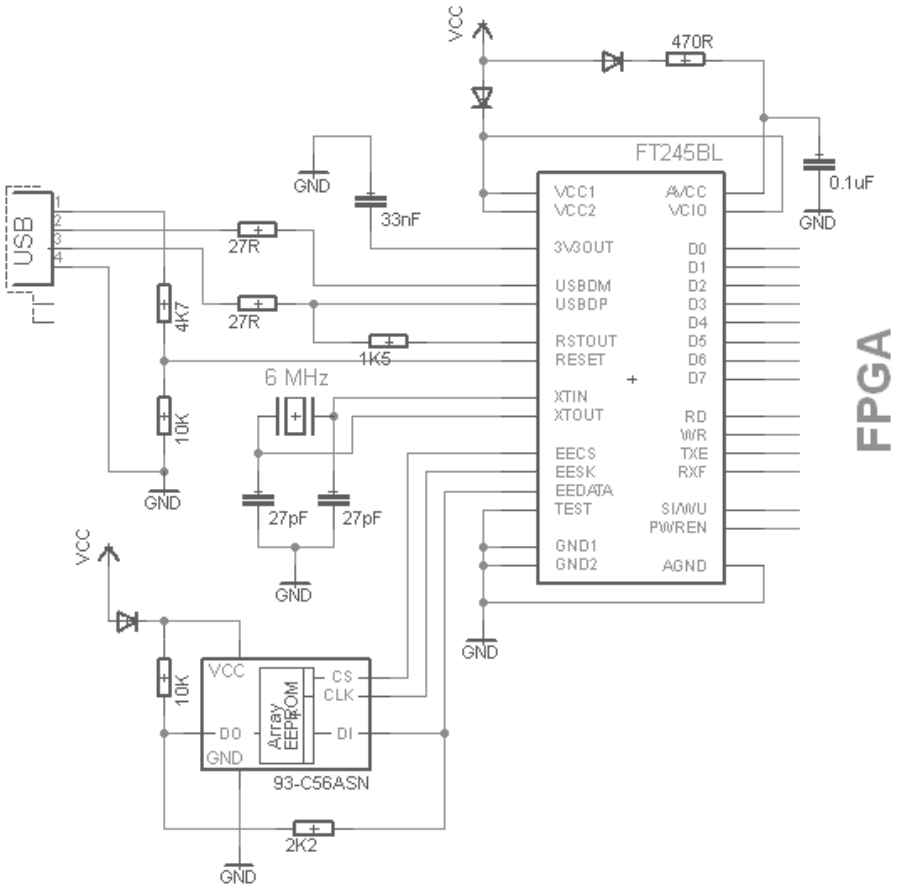
Obr. 3.12: Zesilovač pro LED displej

Samozřejmě je z řídicího obvodu FPGA vyvedeno osm výstupů pro řízení osmi relé ve výkonové části obvodu. Tyto signály nejsou zesilovány, protože v silové části obvodu pouze spínají tranzistory. Podle datasheetu řídicího obvodu Altera MAX EPM7128 (viz přiložené DVD) je tento obvod schopen dodávat maximální proud 25 mA na každý výstupní pin, což je pro spínání relé vyhovující.

Poslední částí schématu je obvod pro USB komunikaci. Protože použitý čip FPGA v sobě nemá přímo implementovaný řadič USB, bylo třeba navrhnout adekvátní náhradu. Tou je paralelní USB převodník FT245BL (viz kapitola 2.3.1.).

3.3.2. Realizace USB komunikace

Paralelní převodník FT245BL je v této aplikaci nejjednodušším možným řešením. Poskytuje řídicímu FPGA obvodu 8 paralelních datových bitů. Na jeden bit tedy připadá jedno spínané relé.



Obr. 3.13: Obvod zajišťující USB komunikaci

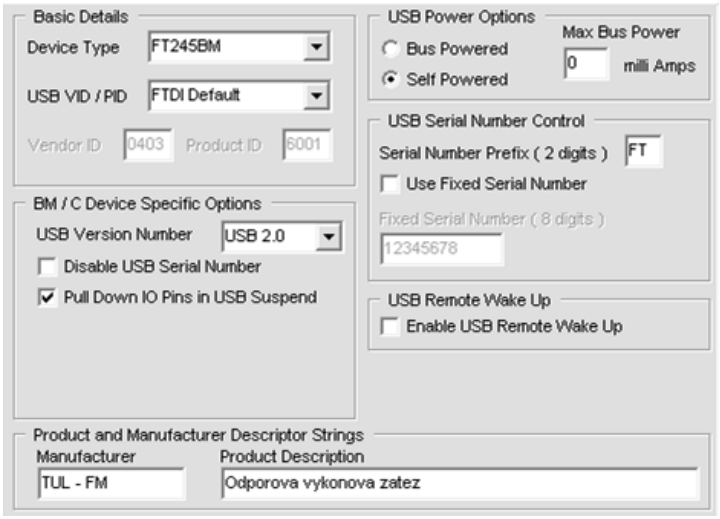
Z celkového počtu 14-ti vstupně/výstupních signálů z FT převodníku do FPGA, obstarává komunikaci řídicí jednotky s PC pouze osm jednosměrně omezených datových signálů (D0 ÷ D7). Zbylých 6 signálů je řídicích, ty řídí komunikaci a synchronizaci, avšak není třeba je používat, protože obvod funguje v BitBang režimu (kapitola 2.3.5.). Datový port obvodu FT je zde pevně nastaven jako výstupní.

V tomto případě není třeba obsluhovat řídicí vstupy a výstupy RD, WR, TXE, RXF, SI/WU. To přenos dat podstatně zjednodušuje. Převodníkem je přímo nastavována binární hodnota odpovídající jedné z 256 hladin elektrického odporu, tuto hodnotu FPGA pouze posílá na dotyčné výstupy pro ovládání relé ve výkonové části obvodu. Pro indikaci připojení PC, jsem použil výstupní signál PWREN, ten je určen

pro řízení externí logiky a udává informaci o aktivním zařízení na USB sběrnici (připojení zařízení k USB).

Nehledě na jednosměrné omezení toku dat je obvod připraven, v případě další modifikace zařízení, pro standardní obousměrný paralelní přenos dat. Navržený obvod to plně umožňuje.

Zapojení obvodu, čili použité externí součástky nutné pro správný chod, jsou použity přesně dle doporučení výrobce (viz datasheet na přiloženém DVD). Do obvodu jsem také aplikoval doporučenou EEPROM paměť 93C56 (kapitola 2.3.2.). Programování paměti EEPROM je možné přímo z řídicí aplikace na PC, nebo speciální utilitou. Já jsem použil speciální aplikaci MProg distribuovanou firmou FTDI.



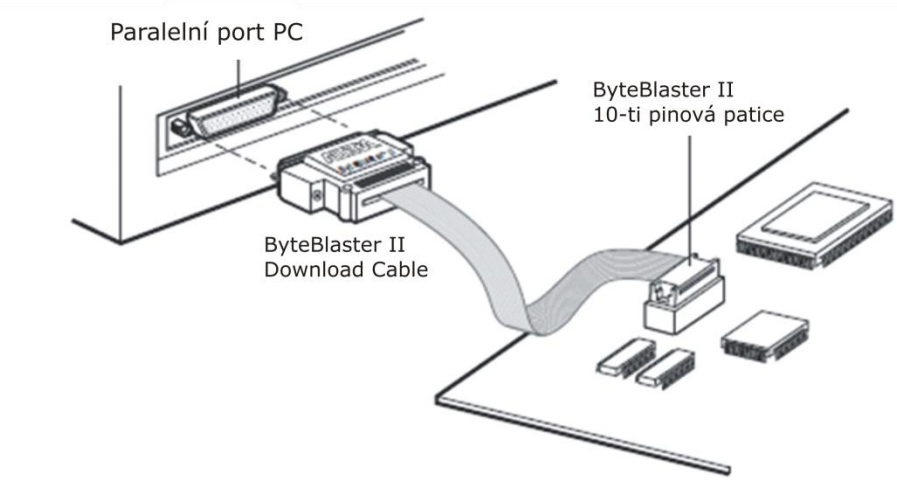
Obr. 3.14: Aplikace MPROG – programování EEPROM

V kapitole 2.2.3 jsou popsány parametry nutné pro enumeraci FT převodníku. Pouze jsem nezmiňoval USB Power Options. Toto nastavení popisuje možnosti napájení převodníku ze sběrnice USB. Je tedy možné zvolit vlastní napájení, nebo napájení ze sběrnice USB, jak je vidět na obrázku (Obr. 3.14) výše. Při volbě možnosti napájení ze sběrnice USB, je nutné specifikovat maximální dovolený odebíraný proud. Protože navržený obvod poskytuje převodníku vlastní napájení, není nutné tento parametr uvádět. V aplikaci je jen matoucí, že umožňuje pouze nastavení obvodu FT245BM. To je však naprosto nepodstatné, protože je až na malé odlišnosti naprosto totožný s obvodem FT245BL (liší se v rychlosti).

Po připojení zařízení s převodníkem FT245BL k PC, je zařízení detekováno operačním systémem s podporou ovladačů D2XX (kapitola 2.3.3.). Celý návod k instalaci je podrobně popsán v příloze (Příloha B).

3.3.3. JTAG rozhraní

Navržený obvod obsahuje port umožňující programování FPGA přímo v obvodu, pomocí tzv. JTAG rozhraní. Zapojení deseti pinů patice pro JTAG, bylo voleno dle programátoru ByteBlasterII firmy Altera (Obr. 3.15). V příloze na DVD je datasheet tohoto programátoru.



Obr. 3.15: Programátor ByteBlasterII firmy Altera

4. Vývoj softwaru

4.1. Práce s knihovnou určenou pro USB převodníky

Knihovna poskytovaná firmou FTDI, pro snadnou komunikaci s převodníkem FT245BL, se skládá z několika souborů DLL a hlavičkového souboru v jazyce C (kapitola 2.3.4.). Do této knihovny se dá přistupovat pomocí souboru ftd2xx.dll, se kterým musí být program staticky či dynamicky sestaven. Jednotlivé funkce, dostupné v této knihovně, mají předpony FT_XX .

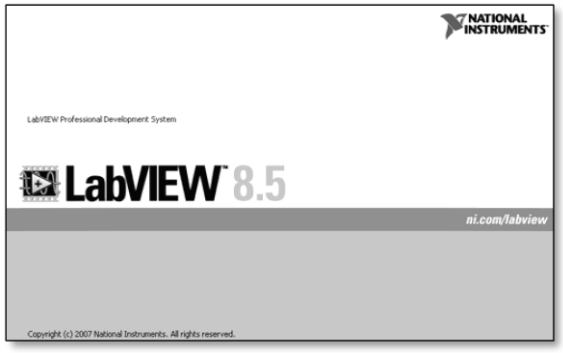
Základní cyklus práce s knihovnou je následující: otevření zařízení, čtení, zápis a uzavření zařízení. Tomu odpovídají jednotlivé funkce FT_Open, FT_Read, FT_Write a FT_Close. Pro otevření je také k dispozici funkce FT_OpenEx, která dokáže otevřít

port nikoliv podle interního číselného identifikátoru, který je nutno zjistit sekvencí volání různých funkcí jako např. FT_ListDevice atd., ale podle textového názvu zařízení připojeného k převodníku.

Ke knihovně ftd2xx.dll lze přistupovat pomocí všech dostupných programovacích jazyků. Na internetových stránkách firmy FTDI (www.ftdi.com) jsou k dispozici jak oba druhy ovladačů, tak i vzorové příklady aplikací přistupujících k USB převodníkům. Na přiloženém DVD jsou tyto příklady samozřejmě obsaženy také. Jde o jednoduché programy psané v jazycích C++, C#, Matlab, LabView, LabWindows, Delphi, Java…a dalších.

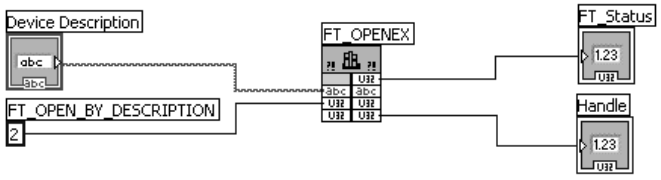
4.2. Počítačová aplikace v prostředí LABVIEW

Pro programování ovládací aplikace zátěže jsem zvolil grafické programovací prostředí LabView 8.5 od firmy National Instruments (Obr. 4.1). Toto prostředí umožňuje snadné a rychlé použití jednotlivých funkcí z knihovny ftd2xx.dll. Samozřejmě umožňuje vytváření spustitelných *.exe souborů pomocí tzv. Application Builderu. Pro běh aplikací vytvořených v LabView, je ale nutné, na počítači kde není LabView nainstalován, doinstalovat volně šiřitelný Labview Runtime Engine (na přiloženém DVD). Poté aplikace fungují bez problémů. Tento engine může být také přímo součástí spustitelného *.exe souboru.



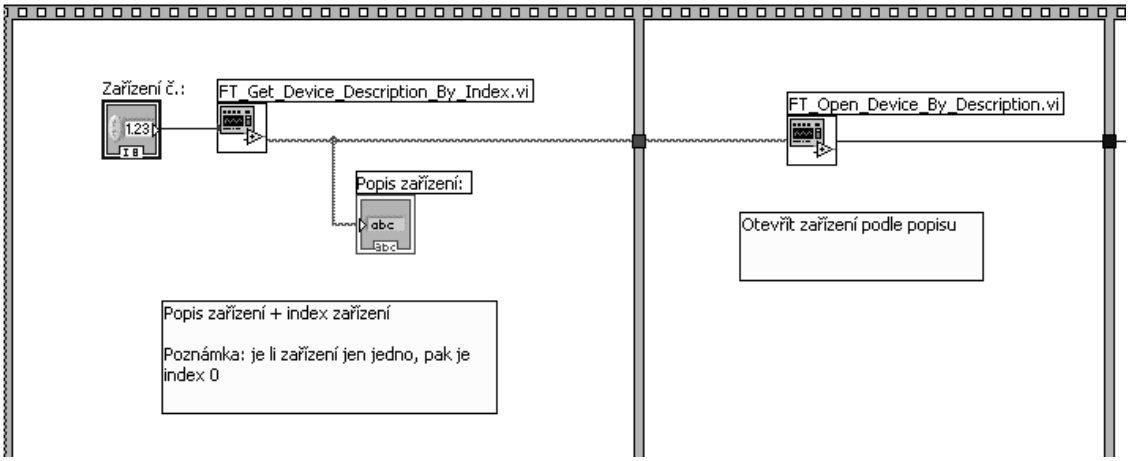
Obr. 4.1: Úvodní okno prostředí LabVIEW

Při návrhu softwaru jsem použil stávající vzorové podprogramy (subVI) z internetových stránek firmy FTDI (www.ftdi.com). Tyto vzorové rutiny jsou na přiloženém DVD. Jejich jednotlivá jména odpovídají funkcím z dynamické knihovny DLL, kterou používají.



Obr. 4.2: Příklad podprogramu pro otevírání zařízení

Na obrázku výše (Obr. 4.2) je pro příklad uveden podprogram pro otevírání zařízení podle jeho popisu. V hlavním VI, tedy hlavním programu se pak jeví funkce takto (Obr. 4.3).



Obr. 4.3: Příklad podprogramu v hlavním programu

Podprogramy v hlavním programu musí pracovat v následující sekvenci:

Nalezení připojeného zařízení - (FT_Get_Device_Description_By_Index.vi) – Obr. 4.3

Otevření vybraného zařízení - (FT_Open_Device_By_description.vi) – Obr. 4.3

Reset zařízení - (FT_Reset_Device.vi)

Nastavení přenosové rychlosti - (FT_Set_Baud_Rate.vi)

Nastavení módu přenosu dat - (FT_Set_Bit Mode.vi) V mém případě BitBang mód. Zde je třeba nastavit tzv. Bit mode masku – touto konstantou se nastavují vstupně/výstupní datové bity dle libosti. Nastavení všech datový bitů na výstupní, například odpovídá konstantě 255. Dále je třeba nastavit Bit mode (typ přenosu dat). BitBang módu odpovídá konstanta 1. Všechny informace o nastavování těchto konstant jsou v programovací příručce knihovny D2XX (viz DVD).

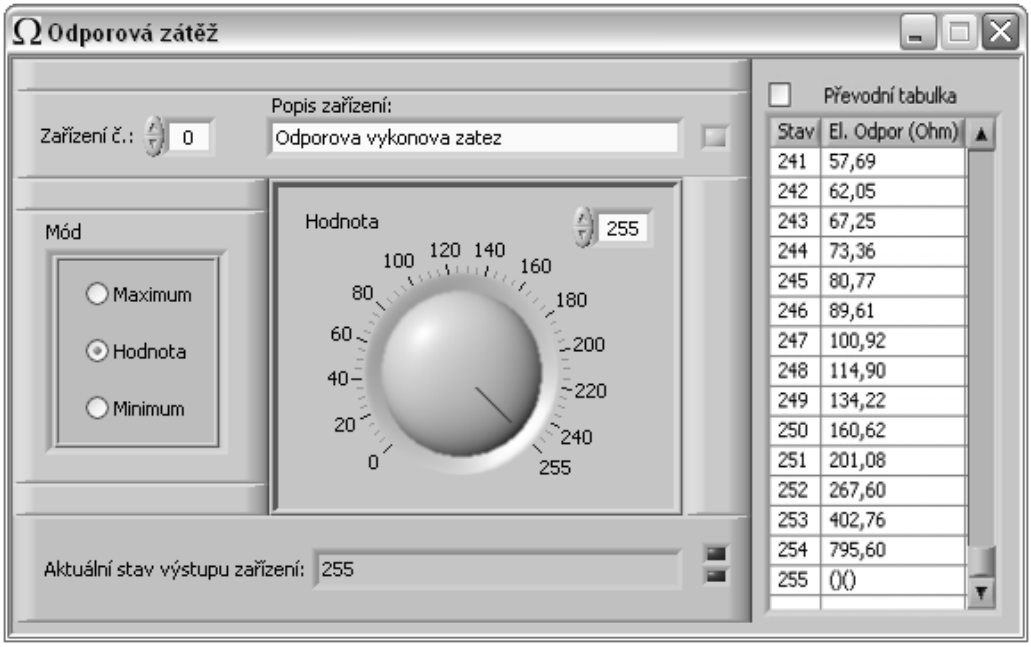
Zápis dat - (FT_Write_Byte_Array_Data.vi) Tímto podprogramem je možno zapisovat data na datové výstupy FT obvodu. Stejně tak by mohlo následovat čtení dat (FT_Read_Byte_Array_data.vi), to však v mé aplikaci nepoužívám.

Zavření zařízení - (FT_Close_Device.vi) Podprogram ukončující komunikaci se zařízením.

Všechny tyto rutiny je nutné zpracovávat cyklicky. Při vykonání každé z nich, se vždy nastavuje tzv. „handle“, čili jakýsi ukazatel na OK, což značí, že funkce byla vykonána správně a lze pokračovat v programu dále.

Při použití jakéhokoliv jiného programovacího jazyka se smysl přístupu k zařízení vůbec neliší. Je nutné vykonat stejné rutinní funkce, pracující s DLL knihovnou FTD2xx.dll.

Grafické uživatelské rozhraní řídicí aplikace pak vypadá takto (Obr. 4.4):



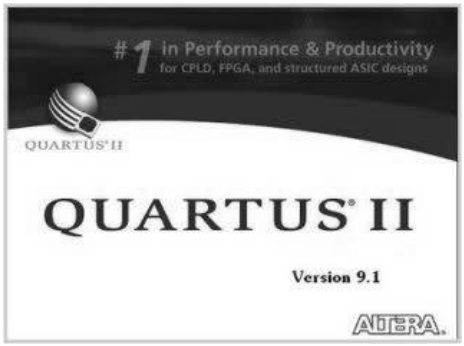
Obr. 4.4: Okno ovládací aplikace odporové zátěže

Application builder při vytváření *.exe souborů umožňuje nastavení stejných detailních funkcí jako jiné vývojové prostředky. Například lze vložit vlastní ikonu spustitelného souboru *.exe, nebo vytvořit instalační program.

Ikona vytvořené aplikace:

4.3. Design VHDL

K programování řídicího FPGA obvodu jsem použil programovací jazyk VHDL (kapitola 2.4.1) a softwarový balík Altera Quartus II 9.1, jenž slouží k přípravě designu na FPGA hradlových polích (kapitola 2.4) firmy Altera.



Obr. 4.5: Úvodní okno prostředí Quartus

Základní prvky VHDL programu jsou procedury:

- Ošetření rotačního enkodéru (kapitola 3.3) – rozlišení směru rotace.
- Nastavení maximálního a minimálního stavu pomocí tlačítek Min. a Max.
- Časovač (1s).
- Čítač odpovídající aktuálnímu stavu výstupu.
- Ošetření USB komunikace.
- Výpis stavů (0 ÷ 255) na LED displej.

VHDL program (Příloha E) jsem vytvářel textovým způsobem programování (kapitola 2.4.1.). Celý zdrojový kód programu je taktéž obsažen na přiloženém DVD a přímo v něm jsou podrobně popsány všechny jeho části a funkce.

5. Shrnutí výsledků práce

5.1. Oživení elektrického obvodu

Oživení obvodů zdroje a silové části probíhalo zcela bez problémů. Při oživování řídicí jednotky jsem zjistil, že převodník připojený na USB sběrnici, napájí celou řídicí jednotku, ačkoliv jsem při nastavování možnosti napájení v paměti EEPROM volil možnost vlastního napájení obvodu. Proto jsem dodatečně osadil do obvodu blokovací diody, které tomuto zamezují. Protože při vypnutém napájení zátěže, by USB sběrnice napájela celou řídicí jednotku, což by mohlo způsobit překročení maximálního dovoleného odběru proudu ze sběrnice (200 ÷ 500 mA).

5.2. Měření výkonových rezistorů

Protože jsem zjistil, že výkonové rezistory nesplňují parametry, které výrobce uvádí (kapitola 1), bylo nutné kompenzovat tyto nepřesnosti.

Tab. 5.1: Skutečné hodnoty rezistorů

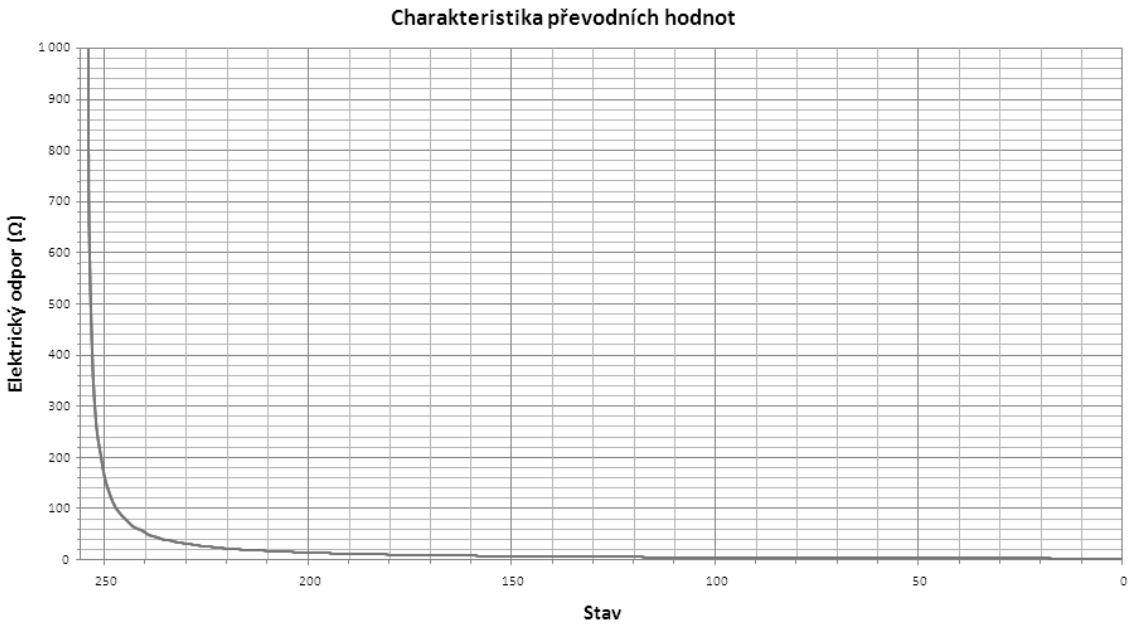
Rezistor	Elektrický odpor (Ω)
R_1	392,84
R_2	402,76
R_3	201,08
R_4	100,92
R_5	46,09
R_6	25,16
R_7	12,69
R_8	6,21

Z těchto důvodů jsem k rezistoru R_5 (46,09 Ω) sériově připojil kompenzační rezistory (4 × 1 Ω) (Příloha E).

K rezistoru R_1 jsem dále připojil sériově čtyři výkonové stoohmové rezistory, čímž k úrovním odporu přibude hodnota 795,60 Ω. Zvýšení impedance tohoto rezistoru hlavně napomůže ke snížení dolní hranice rozsahu impedance zařízení (minimum 3,26 Ω).

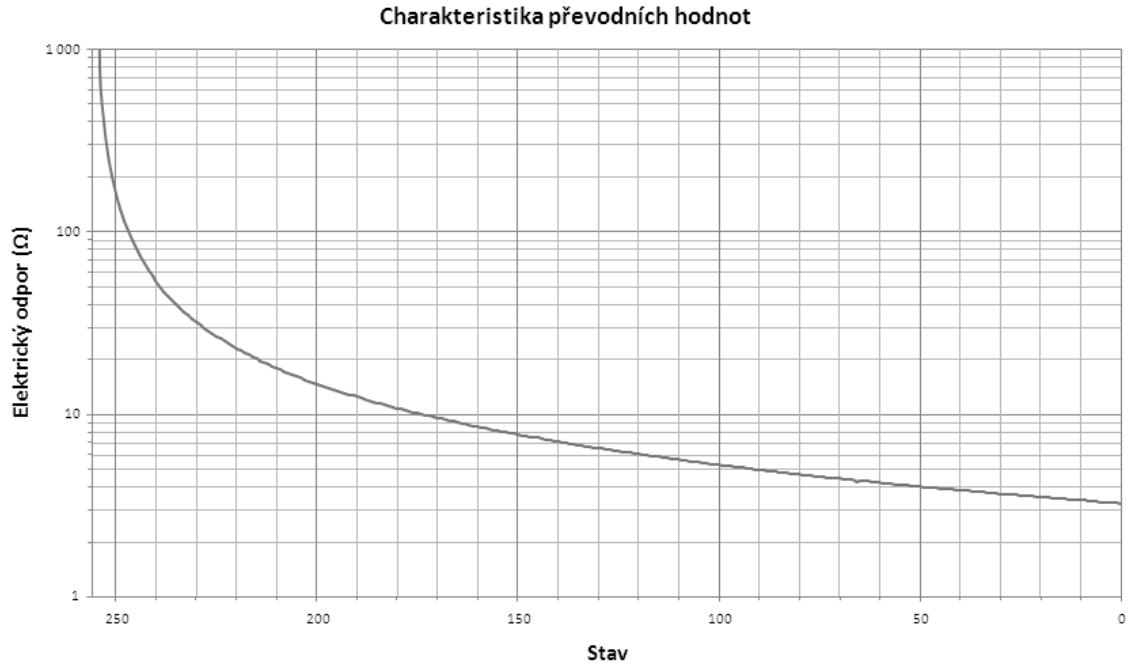
5.3. Měření impedančních úrovní odporové zátěže

V tabulce (Příloha A) jsou hodnoty 256-ti naměřených impedančních úrovní. Toto měření jsem provedl multimetrem Chauvin C.C. 5289 Arnoux (N°945768FCH). V následujících grafech jsou vyneseny závislosti impedance na stavu voleném na displeji ovládacího panelu.



Obr. 5.1: Charakteristika impedančních úrovní v lineárním měřítku

V logaritmickém měřítku pak závislost vypadá takto:



Obr. 5.2: Charakteristika impedančních úrovní v lineárním měřítku

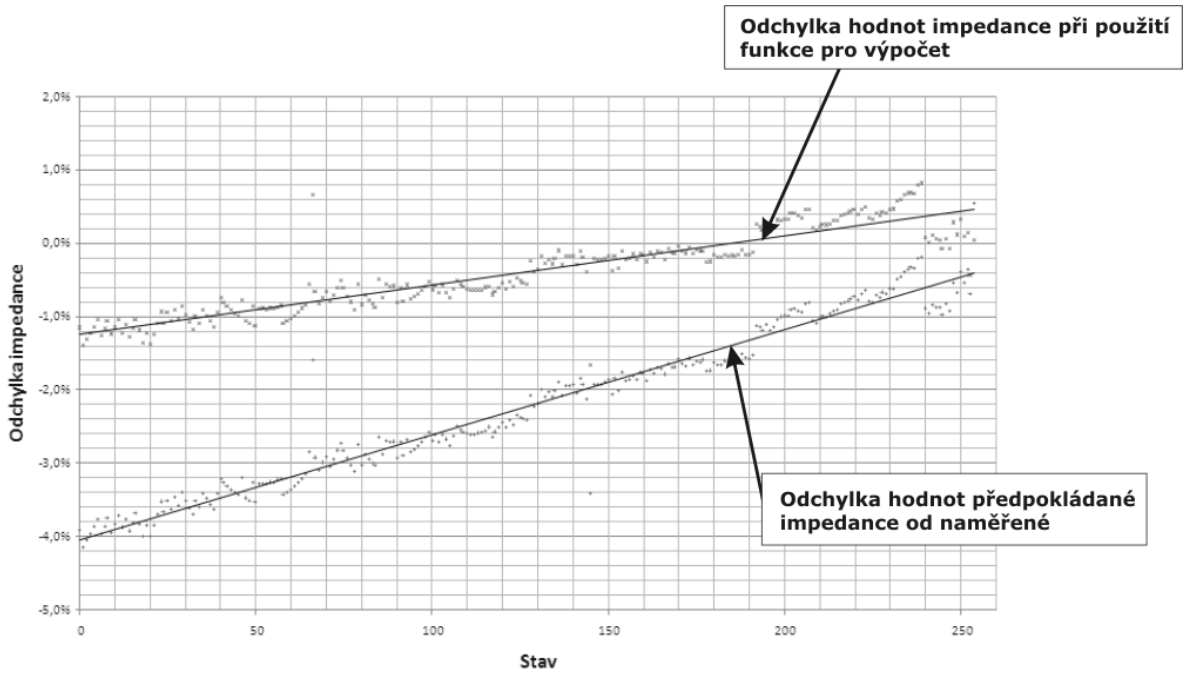
Závislost impedance (Obr. 5.1) je možné přibližně popsat funkcí:

$$R = 798{,}54 \cdot \frac{1}{255-stav} + 0{,}2012 \qquad [5.1]$$

V ideálním případě, kdyby hodnoty impedancí rezistorů byly přesné, by platilo:

$$R = 800 \cdot \frac{1}{255-stav} \qquad [5.2]$$

stav ……. převodní hodnota na displeji



Obr. 5.3: Odchylka hodnot impedance

Na obrázku výše (obr. 5.3) je odchylka hodnot při použití funkce 5.1 pro výpočet impedance a odchylka skutečné hodnoty od předpokládané vypočtené hodnoty dle impedancí výkonových rezistorů (hodnot daných výrobcem v kapitole 1).

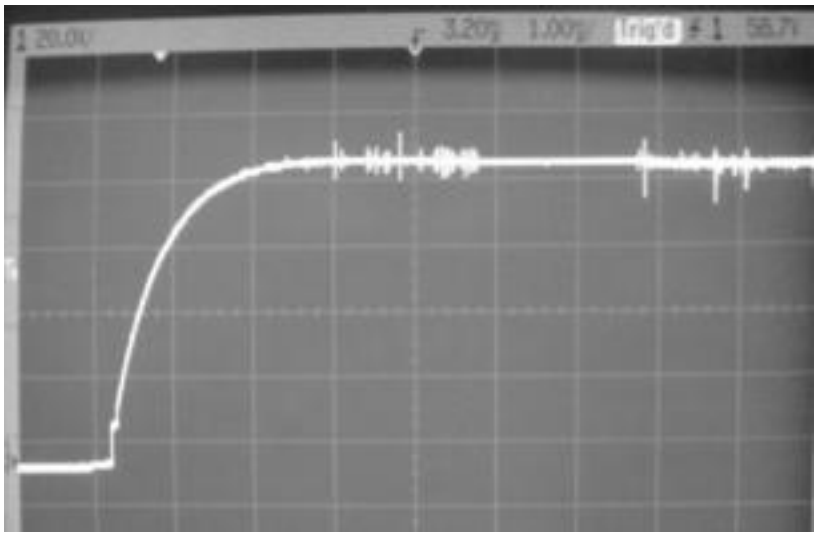
Při spínání rezistorových sítí dochází k nepříjemnému jevu. Celková hodnota impedance paralelní sítě náhle vzroste a poté se vrátí na požadovanou hodnotu. To je důsledkem časové konstanty, která je potřebná k sepnutí relé. Při nastavování spínacích vstupů relé nedochází zároveň k sepnutí a vypnutí příslušných relé. Vypínaná relé se přenastaví dříve, než spínaná. Tento jev bude nutné dodatečně softwarově odstranit ošetřením výstupů řídicího FPGA obvodu. Ale jelikož při kompilaci programu bylo dosaženo 125 ze 128 makrobuněk (98 %), což znamená, že FPGA obvod využívá téměř plně svou kapacitu, další rozšíření programu je velmi problematické. V tomto případě

bych doporučil použít jiný FPGA obvod, nebo snížit taktovací frekvenci na 100 kHz. Např. FPGA Altera EPM 7160SLC84 je totožný, ale obsahuje více makrobuněk (160). Zmíněných 100 kHz by ještě umožnilo použít stávající FPGA. Jelikož by se snížila taktovací frekvence, nebyl by časovač potřebný k ošetření výstupů, z výše uvedených důvodů, tolik početně náročný na makrobuňky. Snížení taktovací frekvence tedy napomůže snížit počet použitých makrobuněk, což je potřebné k úpravě VHDL programu. Úprava programu by pak obsahovala časovou konstantu (řádově milisekundy), při které by bylo ošetřeno nestejnoměrné spínání a rozpínání relé.

Úprava VHDL programu by vypadala takto: Nejprve by se sepnula relé určená k sepnutí a teprve poté by se vypínala relé určená k rozepnutí. Toho lze docílit pouhým využitím logického součtu aktuálního a minulého stavu ovládacích výstupů FPGA, které obstarávají spínání relé. Po dobu zmíněné časové konstanty by na ovládacích výstupech byla tato hodnota logického součtu nastavena (sepnutí všech příslušných relé) a teprve poté by došlo ke změně na konečnou požadovanou hodnotu (vypnutí příslušných relé).

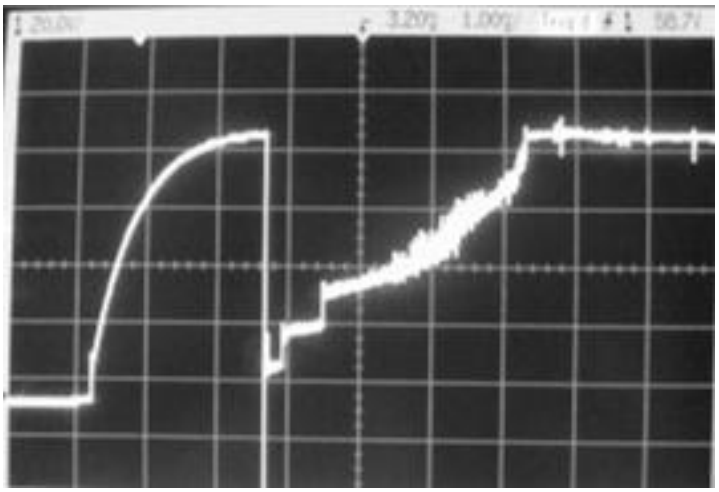
5.4. Ověření funkčnosti zhášecího obvodu

Zhášecí obvod funguje bez problémů. Jeho funkčnost jsem ověřil měřením při 95 V a 15 A stejnosměrného proudu. Na obrázku Obr. 5.4 je průběh napětí na kontaktech relé.



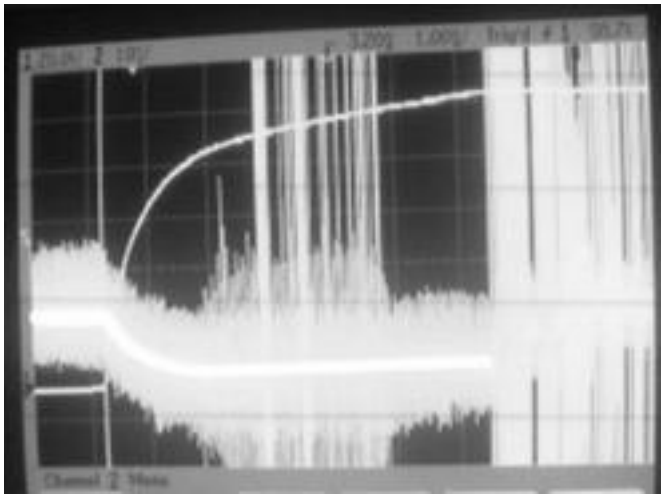
Obr. 5.4: Naměřený průběh stejnosměrného napětí na relé na větvi s rezistorem 6.25 Ohm při napětí 95 V a proudu 15 A, vliv odlehčovacího obvodu, průběh bez jiskření na kontaktech relé

Zhášecí obvody (kapitola 3.1) zaručují při měřených 15 A bezpečné rozepnutí všech relé. Na kontaktech relé je jiskření sice patrné, ale je značně redukované. Zhášecí obvody tedy svou funkci jednoznačně plní. Na obrázku 5.5 je průběh napětí na neošetřených silových kontaktech relé, na nichž dochází k jiskření – hoření el. oblouku.



Obr. 5.5: Naměřený průběh stejnosměrného napětí na jiskřícím relé na větvi s rezistorem

Průběh proudu v silových kontaktech relé při jeho rozpínání (Obr. 5.6)



Obr. 5.6: Naměřený průběh stejnosměrného napětí na relé na větvi s rezistorem 6.25 Ohm při napětí 95 V a proudu 15 A, vliv odlehčovacího obvodu, průběh bez jiskření na kontaktech relé, doplněn průběh proudu snímaný proudovou sondou

6. Závěr

Celkově je možno říci, že cíle kladené v zadání byly splněny a výsledkem této práce je funkční spínaná odporová zátěž.

Byl realizován navržený obvod pro USB komunikaci, jehož jádrem je USB převodník FT245BL a řídicí logický obvod FPGA Altera MAX EPM7128. Bylo vytvořeno uživatelské rozhraní v podobě aplikace programované v prostředí LabView. Aplikace je schopna bezproblémově komunikovat s odporovou zátěží. Z důvodů možných pozdějších modifikací zařízení, byly upraveny a popsány volně distribuované balíky knihoven pro USB převodník FTDI, podporující programovací platformy LabView. V příloze na DVD jsou uvedeny příklady programování aplikací, pracujících s obvodem FT245BL (Delphi, C#, C++ a Matlab). V neposlední řadě byl naprogramován logický obvod (FPGA) v řídicí jednotce v jazyce VHDL.

Spínaná odporová zátěž byla odzkoušena a bylo provedeno měření přesných hodnot elektrického odporu na výstupních svorkách. Nastavená hodnota odporu spínané zátěže vyhovuje plánovaným hodnotám s relativní odchylkou menší, než 4%, při zavedení aproximace s lineární funkcí (viz. funkce 5.1) je výsledná relativní odchylka od nastavené hodnoty menší, než 1,6 %.

Nakonec jsem sestavil podrobný návod k instalaci a obsluze zařízení. (Příloha B a Příloha C).

V případě další modifikace tohoto zařízení, by bylo vhodné redukovat špičkové nárůsty elektrického odporu při spínání odporových sítí (kapitola 5.3). Aby bylo možné použít navrhovanou kompenzaci, bylo by nutné zaměnit FPGA Altera 7128 za ekvivalentní FPGA s více makrobuňkami (minimálně se 150 makrobuňkami), nebo snížit taktovací frekvenci FPGA na 100 kHz. A to z důvodů omezené kapacity obvodu (128 makrobuněk). Dále by bylo vhodné doplnit stávající ovládací panel displejem ampérmetru, jenž by zobrazoval aktuální protékaný proud. Stejně tak by tato hodnota mohla být zobrazována v ovládací aplikaci na PC. To by si samozřejmě také vyžádalo jistou úpravu řídicího obvodu, který by se rozrostl o A/D převodník. Použitý USB převodník umožňuje obousměrnou komunikaci a nabízí tak neomezené možnosti rozšíření funkcí zařízení.

7. Použitá literatura

[1] Altera Corporation: ByteBlaster II Download Cable. Verze 8.0. 2008 [online]. [cit. 10.5.2011] URL: < http://www.altera.com/literature/ug/ug_bbii.pdf>

[2] *FT245BM Designers Guide*. Verze 1.1. Future Technology Devices International, 2002 [online]. [cit. 11.4.2011] URL: < http:// www.ftdichip.com>

[3] *FTD2XX Programmers Guide*. Verze 2.01. Future Technology Devices International, 2002[online]. [cit. 11.5.2011] URL: < http:// www.ftdichip.com>

[4] Hw.cz.: *USB 2.0*[online]. [cit. 11.5.2011] URL: < http://hw.cz/Rozhrani/ART1232-USB-2.0---dil-1.html>

[5] Kolouch, Jaromír: *Programovatelné logické obvody a hradlová pole – moderní stavební prvky číslicových systémů*. Automatizace, ročník 52, číslo 1 leden 2009

[6] Kůs,V.,Skála,J.: Elektronika a elektrické pohony pro neelektrotechnické fakulty. Skriptum ZČU Plzeň, 1998

[7] Linhart, J. a Turek, M.: *Topologie USB*. [online]. [cit. 11.5.2011]
URL: < http://phxweb.wz.cz/usb/index.php?menu=1&pol=1&kat=3&sublev=2>

[8] *Universal Serial Bus Specification Revision 2.0*. USB Implementers Forum, 2000[online]. [cit. 11.5.2011] URL: < http:// www.usb.org>

[9] Kainka, Burkhard: *USB - měření, řízení a regulace pomocí sběrnice USB - základy, popis a charakteristika rozhraní USB, mikrokontroléry*. BEN - technická literatura, 2002, ISBN 80-7300-073-3

[10] Chroma Systems Solutions, Inc.: *Programmable AC&DC Electronic load* [online]. [cit. 11.5.2011] URL: <http://www.chromausa.com/pdf/63800-EDM1.pdf>

[11] Kolouch, J.: *Programovatelné logické obvody a návrh jejich aplikací v jazyku VHDL*. Skriptum FEKT VUT v Brně. Brno, 2006.

[12] Matoušek,D. : *USB prakticky s obvody FTDI*. BEN – technická literatura, Praha 2003, ISBN: 80-7300-103-9

[13] Pech, Jan: Programovatelné logické obvody [online]. [cit. 11.5.2011] URL:<http://fpga.sweb.cz/>

[14] Poupa, Martin: *Přednášky z předmětu Programovatelné logické obvody*. Fakulta elektrotechnická, Západočeská univerzita v Plzni [online]. [cit. 11.5.2011]
URL: <http:// www.vyuka.fel.czu.cz/kae/plo/plo_prednasky.pdf>

[15] Syrový, Milan: *Semestrální projekt – Návrh a realizace elektronického obvodu pro řízení lineární odporové zátěže pro laboratoř elektrických pohonů.* Technická univerzita v Liberci, Liberec 2010.

[16] *Katalogové listy použitých součástek*

8. Obsah přiloženého DVD

Adresář	Obsah adresáře
\DataSheet	Katalogové listy použitých součástek
\Manuály	Návod k instalaci a obsluze
\USB ovladače	Ovladače pro USB (D2XX)
\Run time engine 8.5	Engine potřebný pro chod aplikace
\Schémata+DPS	
-\Řídicí část	Schéma a DPS řídicí části
-\Výkonová část	Schéma a DPS výkonové části
- \Zdroj napětí	Schéma a DPS napájecího obvodu
\Software_VHDL	VHDL kód programu pro FPGA
\Software_PC	Ovládací aplikace vytvořená v LabView
\Text_DP	Text diplomové práce ve formátu pdf
\Ostatní	Knihovny funkcí FTDI pro různé programovací a výpočetní platformy, programovací příručka pro knihovnu FTD2xx.dll, …

9. Přílohy

Převodní tabulka

Stav	El. odpor(Ω)	Stav	El. odpor(Ω)	Stav	El. odpor(Ω)	Stav	El. odpor(Ω)	Stav	El. odpor(Ω)	Stav	El. odpor(Ω)	Stav	El. odpor(Ω)	Stav	El. odpor(Ω)
255	∞	223	25,16	191	12,69	159	8,48	127	6,40	95	5,14	63	4,30	31	3,70
254	795,60	222	24,41	190	12,50	158	8,40	126	6,35	94	5,11	62	4,28	30	3,68
253	402,76	221	23,71	189	12,31	157	8,31	125	6,30	93	5,07	61	4,26	29	3,66
252	267,60	220	23,02	188	12,12	156	8,23	124	6,25	92	5,05	60	4,24	28	3,65
251	201,08	219	22,39	187	11,95	155	8,15	123	6,21	91	5,01	59	4,22	27	3,63
250	160,62	218	21,79	186	11,78	154	8,06	122	6,16	90	4,99	58	4,20	26	3,62
249	134,22	217	21,22	185	11,61	153	8,00	121	6,12	89	4,95	57	4,17	25	3,60
248	114,90	216	20,68	184	11,45	152	7,91	120	6,07	88	4,92	56	4,15	24	3,59
247	100,92	215	20,18	183	11,29	151	7,85	119	6,03	87	4,89	55	4,13	23	3,57
246	89,61	214	19,69	182	11,14	150	7,76	118	5,99	86	4,87	54	4,11	22	3,56
245	80,77	213	19,23	181	10,99	149	7,69	117	5,95	85	4,83	53	4,09	21	3,55
244	73,36	212	18,79	180	10,84	148	7,62	116	5,90	84	4,82	52	4,07	20	3,54
243	67,25	211	18,36	179	10,71	147	7,55	115	5,86	83	4,79	51	4,05	19	3,52
242	62,05	210	17,96	178	10,57	146	7,48	114	5,82	82	4,76	50	4,04	18	3,51
241	57,69	209	17,58	177	10,42	145	7,42	113	5,78	81	4,73	49	4,01	17	3,49
240	53,81	208	17,20	176	10,29	144	7,36	112	5,74	80	4,71	48	4,00	16	3,47
239	50,09	207	16,80	175	10,16	143	7,28	111	5,70	79	4,67	47	3,98	15	3,46
238	47,15	206	16,46	174	10,04	142	7,21	110	5,66	78	4,66	46	3,95	14	3,45
237	44,59	205	16,15	173	9,91	141	7,16	109	5,62	77	4,63	45	3,94	13	3,43
236	42,24	204	15,83	172	9,80	140	7,09	108	5,58	76	4,60	44	3,92	12	3,42
235	40,14	203	15,52	171	9,68	139	7,03	107	5,54	75	4,57	43	3,90	11	3,40
234	38,24	202	15,23	170	9,56	138	6,97	106	5,51	74	4,54	42	3,88	10	3,39
233	36,53	201	14,96	169	9,46	137	6,92	105	5,48	73	4,52	41	3,86	9	3,38
232	34,95	200	14,69	168	9,35	136	6,85	104	5,44	72	4,50	40	3,84	8	3,36
231	33,54	199	14,43	167	9,24	135	6,80	103	5,40	71	4,48	39	3,83	7	3,35
230	32,20	198	14,18	166	9,15	134	6,75	102	5,37	70	4,45	38	3,82	6	3,34
229	30,98	197	13,95	165	9,04	133	6,69	101	5,33	69	4,43	37	3,80	5	3,32
228	29,83	196	13,72	164	8,94	132	6,64	100	5,30	68	4,41	36	3,78	4	3,31
227	28,78	195	13,48	163	8,85	131	6,58	99	5,26	67	4,38	35	3,76	3	3,30
226	27,78	194	13,27	162	8,75	130	6,54	98	5,23	66	4,36	34	3,75	2	3,29
225	26,88	193	13,05	161	8,67	129	6,49	97	5,20	65	4,33	33	3,73	1	3,28
224	26,01	192	12,84	160	8,57	128	6,43	96	5,17	64	4,32	32	3,72	0	3,26

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

VÝKONOVÁ ODPOROVÁ ZÁTĚŽ

Instalační příručka

Instalační příručka pro Windows XP

Ovladače

Ovladače jsou dostupné na:

http://www.ftdichip.com/Drivers/D2XX.htm
http://www.ftdichip.com/Drivers/VCP.htm

Pro OS Windows XP/Vista/7 jsou k dispozici dva druhy ovladačů – přímé ovladače (D2XX) a ovladače VCP (Virtual COM Port). Při instalaci se přidá do systému nový COM port, na který se potom přistupuje stejně jako ke všem ostatním COM portům přes Windows API nebo přímo. Existují verze s podporou plug & play a bez ní. Druh ovladačů je třeba volit dle řídicí aplikace.

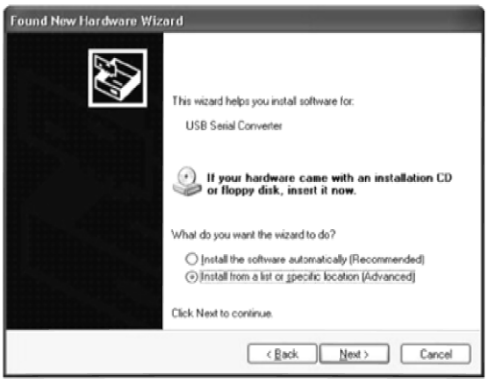
Instalace ve Windows XP

Zapněte počítač a propojte USB kabelem PC a Odporovou zátěž. Zobrazí se následující okno:

Vyberte volbu „Instalovat ze seznamu či daného umístění“ a stiskněte tlačítko „Další“.

2

Instalační příručka pro Windows XP



Vyberte adresář, ve kterém máte umístěn D2XX ovladač (například D:/ovladače/) a potvrďte pomocí tlačítka „Ok."

Zvolený adresář je vypsán v dialogovém okně. Stiskněte tlačítko „Další".

3

Instalační příručka pro Windows XP

Pokračujte tlačítkem „Pokračovat“.

Ovladače konvertoru se nainstalují.

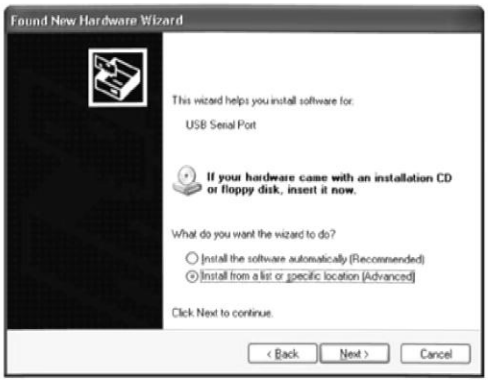
Nakonec potvrďte „Dokončit“, instalace se dokončí.

4

Instalace konvertoru je hotová. V dalším kroku je nutné nainstalovat virtuální COM port.

Instalační program ovladačů virtuálního COM portu se spustí ihned po ukončení instalace USB sériového konvertoru.

Je nutné postupovat jako v předchozím případě. Vybrat možnost „Instalovat ze seznamu či daného umístění“ a stisknout tlačítko „Další“.



Vybrat adresář obsahující ovladače a pokračovat tlačítkem „Další“.

5

Instalační příručka pro Windows XP

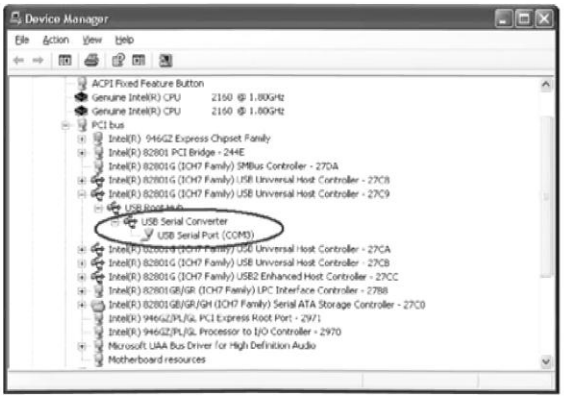
Potvrdit pokračování „Pokračovat“.

6

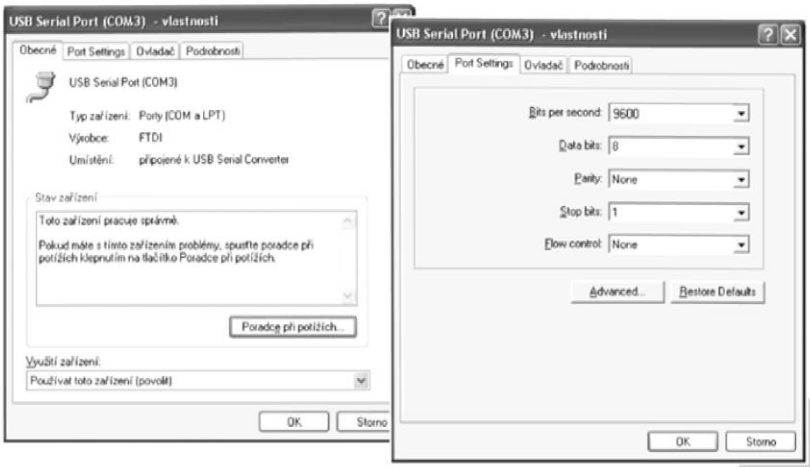
Instalační příručka pro Windows XP

Zbývá potvrdit dokončení instalace a vypnout instalační program tlačítkem „Dokončit“.

Po této instalaci je ve správci zařízení nainstalované zařízení již přístupné.

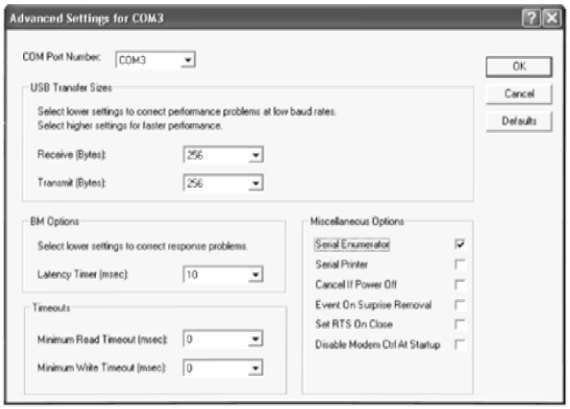


Poté můžeme nastavovat jeho vlastnosti a parametry.



7

Instalační příručka pro Windows XP



Nastavení parametrů:

- Rychlost komunikace 9600 bit/s
- Přenos 8 bitový přenos bez parity
- 1 stop bit
- Flow kontrol NONE
- Port COM 3 (nebo první volný)
- FIFO buffer s rychlostí RX-256 a TX-256
- Timeout 10 ms

Instalační příručky pro jiné operační systémy jsou dostupné na:

http://www.ftdichip.com/Support/Documents/InstallGuides.htm

V Liberci **2011**

8

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

VÝKONOVÁ ODPOROVÁ ZÁTĚŽ

Návod k obsluze

Návod k obsluze zařízení

Odporová zátěž (dále jen zátěž) může pracovat ve dvou režimech.

- Autonomní režim – zařízení je ovládáno uživatelem z ovládacího panelu zařízení.
- Režim dálkového ovládání – zařízení je ovládáno řídicí aplikací z PC.

Práce s ovládacím panelem

Po zapnutí přístroje se spustí chladící ventilátory a elektrický odpor (dále jen odpor) mezi výstupními svorkami zátěže se blíží k nekonečnu. Na displeji ovládacího panelu (Obr. 1) se objeví stav 255. Zařízení je připraveno k použití.

Obr. 1: Ovládací panel přístroje

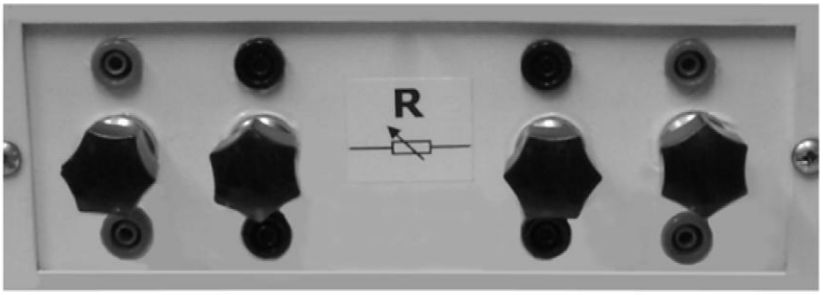
- Uživatel může pomocí „točítka" na ovládacím panelu libovolně měnit úrovně odporu podle převodní tabulky, která je v zadní části návodu nebo na horním krytu zátěže.

- Hodnota odporu na výstupních svorkách přístroje se nemění zároveň s hodnotou na displeji, ale je nastavena po uplynutí cca. 1 sekundy od poslední změny hodnoty na displeji.

- Dále je možné použít rychlou volbu maximální nebo minimální hodnoty pomocí tlačítek Min. a Max. Při použití této volby je nastavena hodnota odpovídajícího odporu ihned, protože tato tlačítka mohou být použita v nouzovém případě, kdy je třeba získat rychle jednu z mezních hodnot.

2

Návod k obsluze zařízení



Obr. 2: Výstupní svorky zátěže

Všech šest výstupních svorek zařízení (Obr. 2) v levé části je navzájem propojených, stejně tak i šest svorek v pravé části. Oba uzly těchto svorek obsahují dvě výkonové svorky pro připojení zatěžovaných aplikací a čtyři menší svorky například pro připojení měřících přístrojů.

Ovládání pomocí PC

Tento přístroj umožňuje komunikaci s PC pomocí USB sběrnice. Zařízení je kompatibilní s USB 1.0 a USB 2.0.

Při připojování nebo odpojování přívodního USB kabelu se doporučuje, aby zařízení bylo ve vypnutém stavu.

- Po připojení USB kabelu a zapnutí přístroje mají výstupní svorky zátěže opět impedanci blížící se k nekonečnu a na displeji svítí nápis PC.

Řídicí aplikace

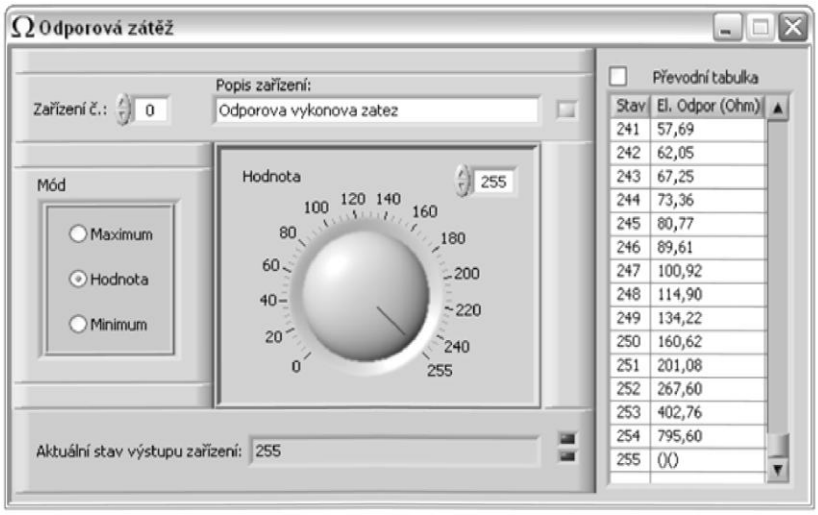
Řídicí aplikaci není nutné instalovat, stačí otevřít její spouštěcí *.exe soubor.

Odporová zátěž.exe
Technická univerzita v Liberci

3

Návod k obsluze zařízení

Aplikace ovšem vyžaduje, aby byl na PC nainstalován volně dostupný LabView Runtime Engine 8.5, nebo celé vývojové prostředí LabView.



obr. 3: Řídicí aplikace

Po spuštění aplikace Odporová zátěž se objeví toto okno (obr. 3). Je-li zařízení připojeno správně a jsou-li správně nainstalovány potřebné ovladače dle instalační příručky, v okénku „Popis zařízení" se objeví popis připojeného zařízení.

Aplikace je připravena ovládat více obdobných zařízení připojených k PC, a proto je v levém horním rohu číslo připojeného zařízení (defaultní hodnota je 0) a v již zmíněném okénku se uvádí jeho popis.

Výstupní odpor zařízení je možné zadávat ve třech režimech:

- Maximum a Minimum – dojde ihned k nastavení odporu na maximální nebo minimální hodnotu.
- Režim potenciometr – pro ovládání tímto způsobem musí být v módu nastavena možnost „Hodnota". Hodnota odporu na výstupu se klasicky mění po 1 s, stejně jako při ovládání „točítkem" na řídícím panelu.

4

- Převodní tabulka – v tomto režimu ovládání lze hodnoty volit přímo v převodní tabulce. Hodnota výstupního odporu se mění opět po 1 s. Pro tento režim volby odporu je nutné zaškrtnout v pravém horním rohu okna možnost „Převodní tabulka".

! V PC režimu ovládání nelze měnit hodnoty pomocí ovládacího panelu.

Převodní tabulka

Stav	El. odpor(Ω)	Stav	El. odpor(Ω)	Stav	El. odpor(Ω)	Stav	El. odpor(Ω)	Stav	El. odpor(Ω)	Stav	El. odpor(Ω)	Stav	El. odpor(Ω)
255	∞	223	25,16	191	12,69	159	8,48	127	6,40	95	5,14	63	4,30	31	3,70
254	795,60	222	24,41	190	12,50	158	8,40	126	6,35	94	5,11	62	4,28	30	3,68
253	402,76	221	23,71	189	12,31	157	8,31	125	6,30	93	5,07	61	4,26	29	3,66
252	267,60	220	23,02	188	12,12	156	8,23	124	6,25	92	5,05	60	4,24	28	3,65
251	201,08	219	22,39	187	11,95	155	8,15	123	6,21	91	5,01	59	4,22	27	3,63
250	160,62	218	21,79	186	11,78	154	8,06	122	6,16	90	4,99	58	4,20	26	3,62
249	134,22	217	21,22	185	11,61	153	8,00	121	6,12	89	4,95	57	4,17	25	3,60
248	114,90	216	20,68	184	11,45	152	7,91	120	6,07	88	4,92	56	4,15	24	3,59
247	100,92	215	20,18	183	11,29	151	7,85	119	6,03	87	4,89	55	4,13	23	3,57
246	89,61	214	19,69	182	11,14	150	7,76	118	5,99	86	4,87	54	4,11	22	3,56
245	80,77	213	19,23	181	10,99	149	7,69	117	5,95	85	4,83	53	4,09	21	3,55
244	73,36	212	18,79	180	10,84	148	7,62	116	5,90	84	4,82	52	4,07	20	3,54
243	67,25	211	18,36	179	10,71	147	7,55	115	5,86	83	4,79	51	4,05	19	3,52
242	62,05	210	17,96	178	10,57	146	7,48	114	5,82	82	4,76	50	4,04	18	3,51
241	57,69	209	17,58	177	10,42	145	7,42	113	5,78	81	4,73	49	4,01	17	3,49
240	53,81	208	17,20	176	10,29	144	7,36	112	5,74	80	4,71	48	4,00	16	3,47
239	50,09	207	16,80	175	10,16	143	7,28	111	5,70	79	4,67	47	3,98	15	3,46
238	47,15	206	16,46	174	10,04	142	7,21	110	5,66	78	4,66	46	3,95	14	3,45
237	44,59	205	16,15	173	9,91	141	7,16	109	5,62	77	4,63	45	3,94	13	3,43
236	42,24	204	15,83	172	9,80	140	7,09	108	5,58	76	4,60	44	3,92	12	3,42
235	40,14	203	15,52	171	9,68	139	7,03	107	5,54	75	4,57	43	3,90	11	3,40
234	38,24	202	15,23	170	9,56	138	6,97	106	5,51	74	4,54	42	3,88	10	3,39
233	36,53	201	14,96	169	9,46	137	6,92	105	5,48	73	4,52	41	3,86	9	3,38
232	34,95	200	14,69	168	9,35	136	6,85	104	5,44	72	4,50	40	3,84	8	3,36
231	33,54	199	14,43	167	9,24	135	6,80	103	5,40	71	4,48	39	3,83	7	3,35
230	32,20	198	14,18	166	9,15	134	6,75	102	5,37	70	4,45	38	3,82	6	3,34
229	30,98	197	13,95	165	9,04	133	6,69	101	5,33	69	4,43	37	3,80	5	3,32
228	29,83	196	13,72	164	8,94	132	6,64	100	5,30	68	4,41	36	3,78	4	3,31
227	28,78	195	13,48	163	8,85	131	6,58	99	5,26	67	4,38	35	3,76	3	3,30
226	27,78	194	13,27	162	8,75	130	6,54	98	5,23	66	4,36	34	3,75	2	3,29
225	26,88	193	13,05	161	8,67	129	6,49	97	5,20	65	4,33	33	3,73	1	3,28
224	26,01	192	12,84	160	8,57	128	6,43	96	5,17	64	4,32	32	3,72	0	3,26

Návod k obsluze zařízení

Napájení

~ 230 V/ max. 3 A

Zatěžovací parametry

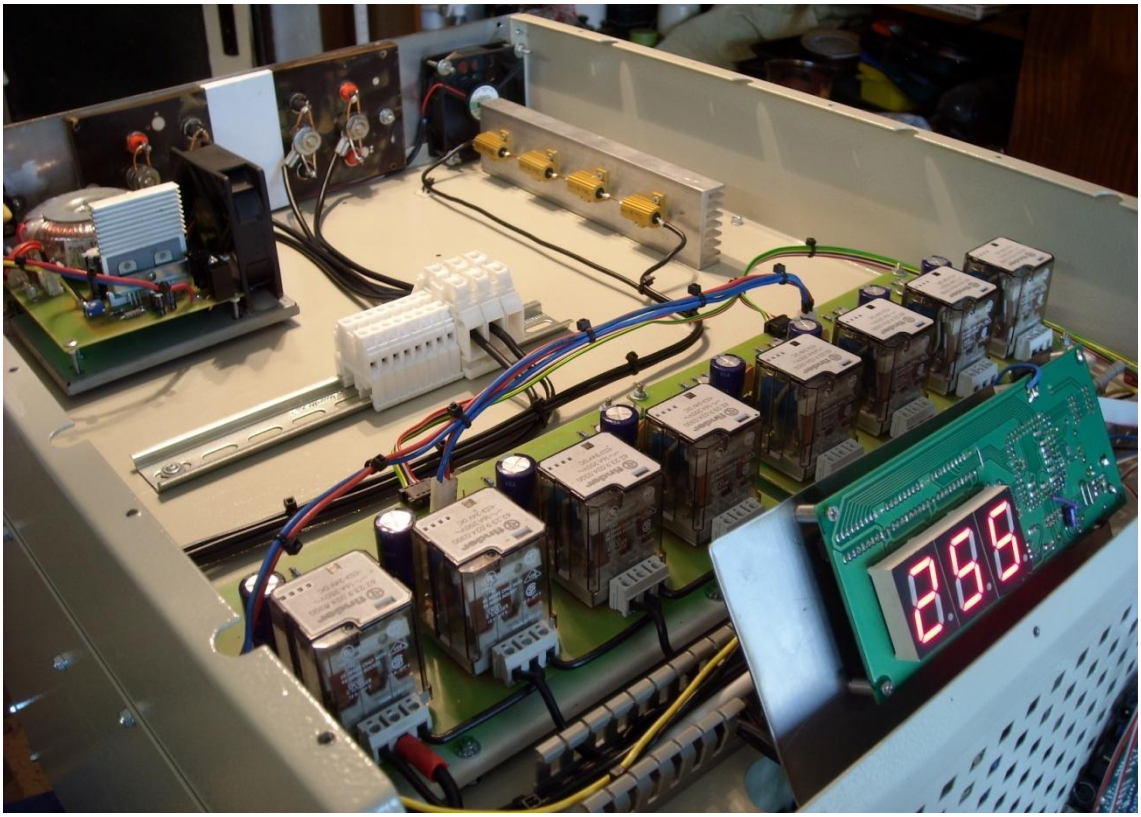
Maximální trvalý ztrátový výkon	2500 W
Maximální ss. napětí	250 V
Maximální ss. proud	15 A

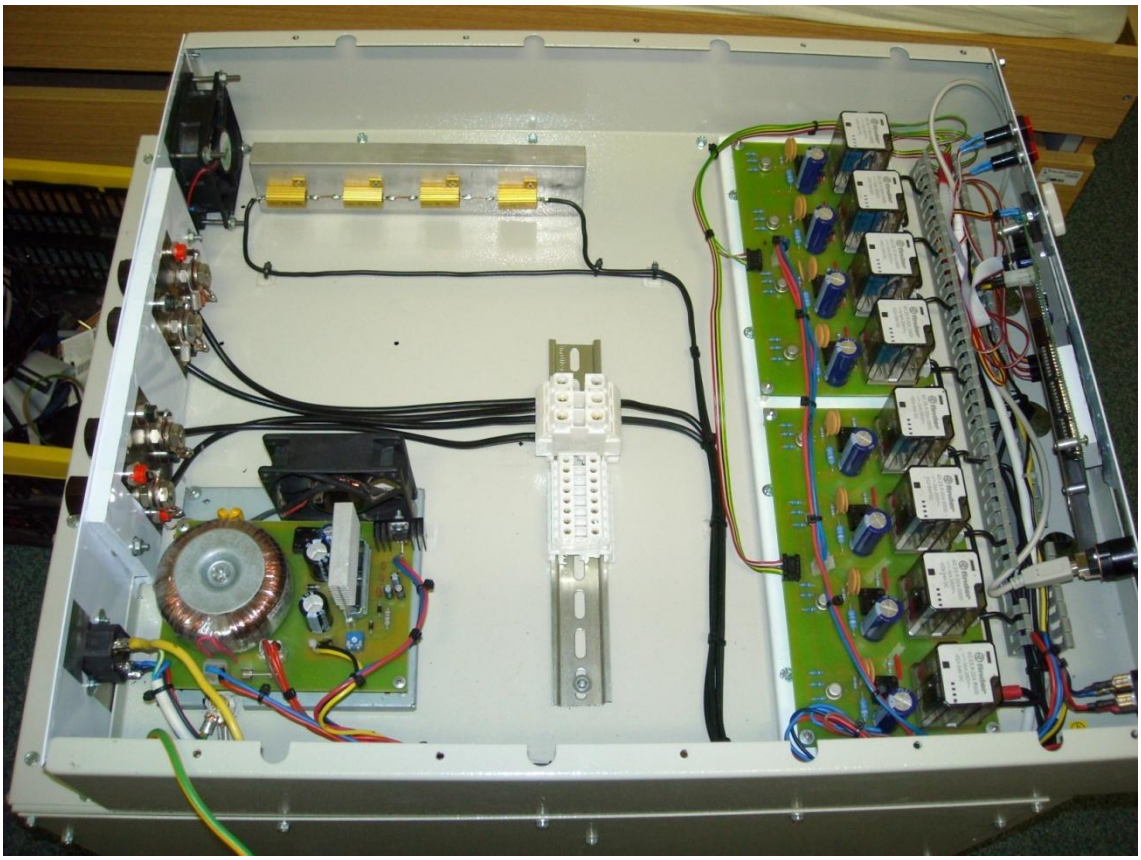
LabView Runtime Engine 8.5 je volně dostupný na:

http://www.ni.com/support/

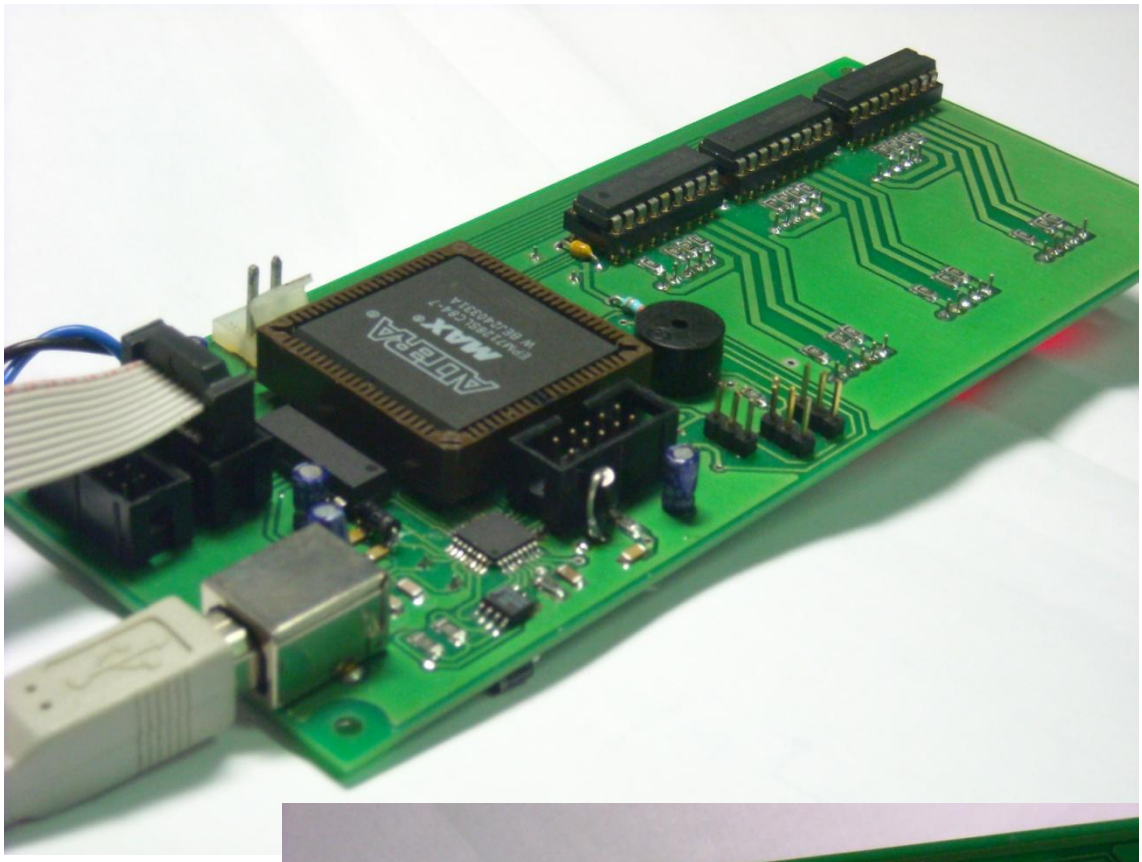
V Liberci **2011**

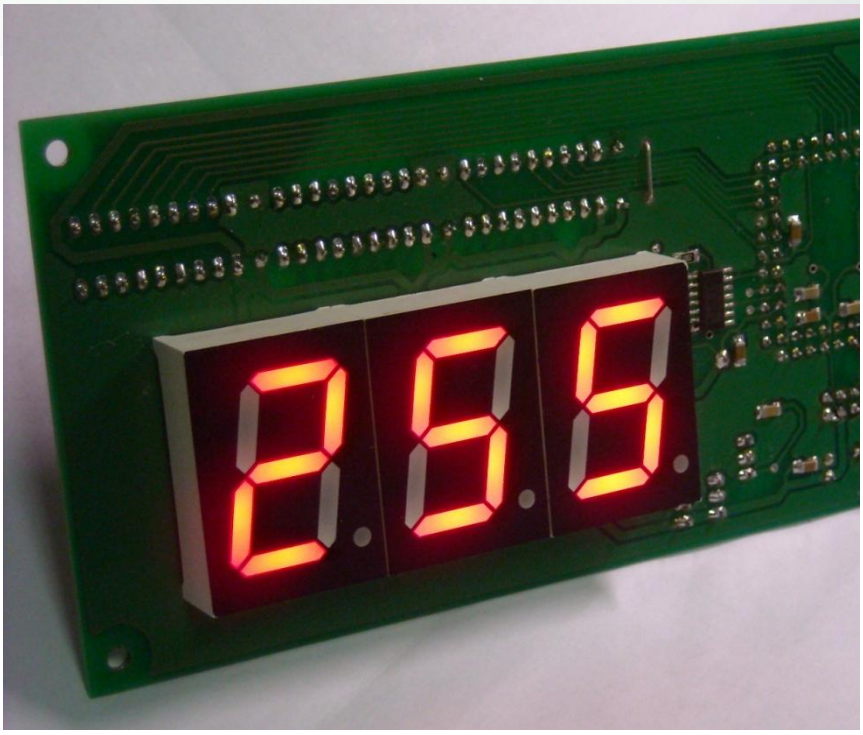
7



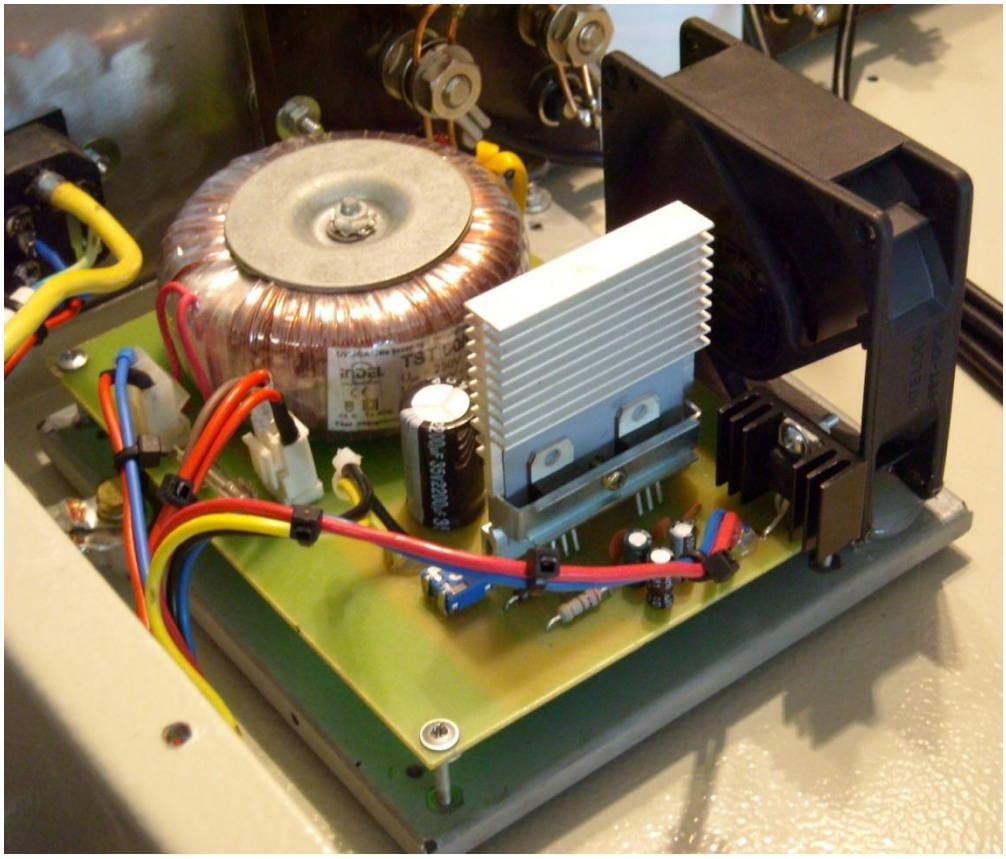


Řídicí jednotka

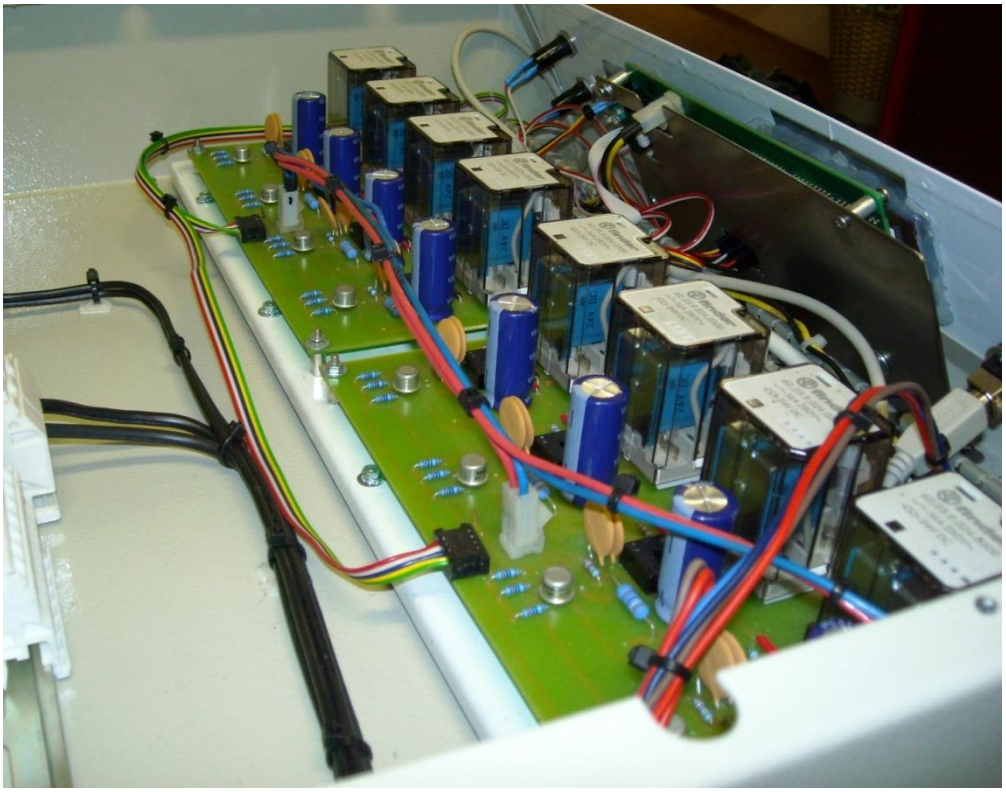




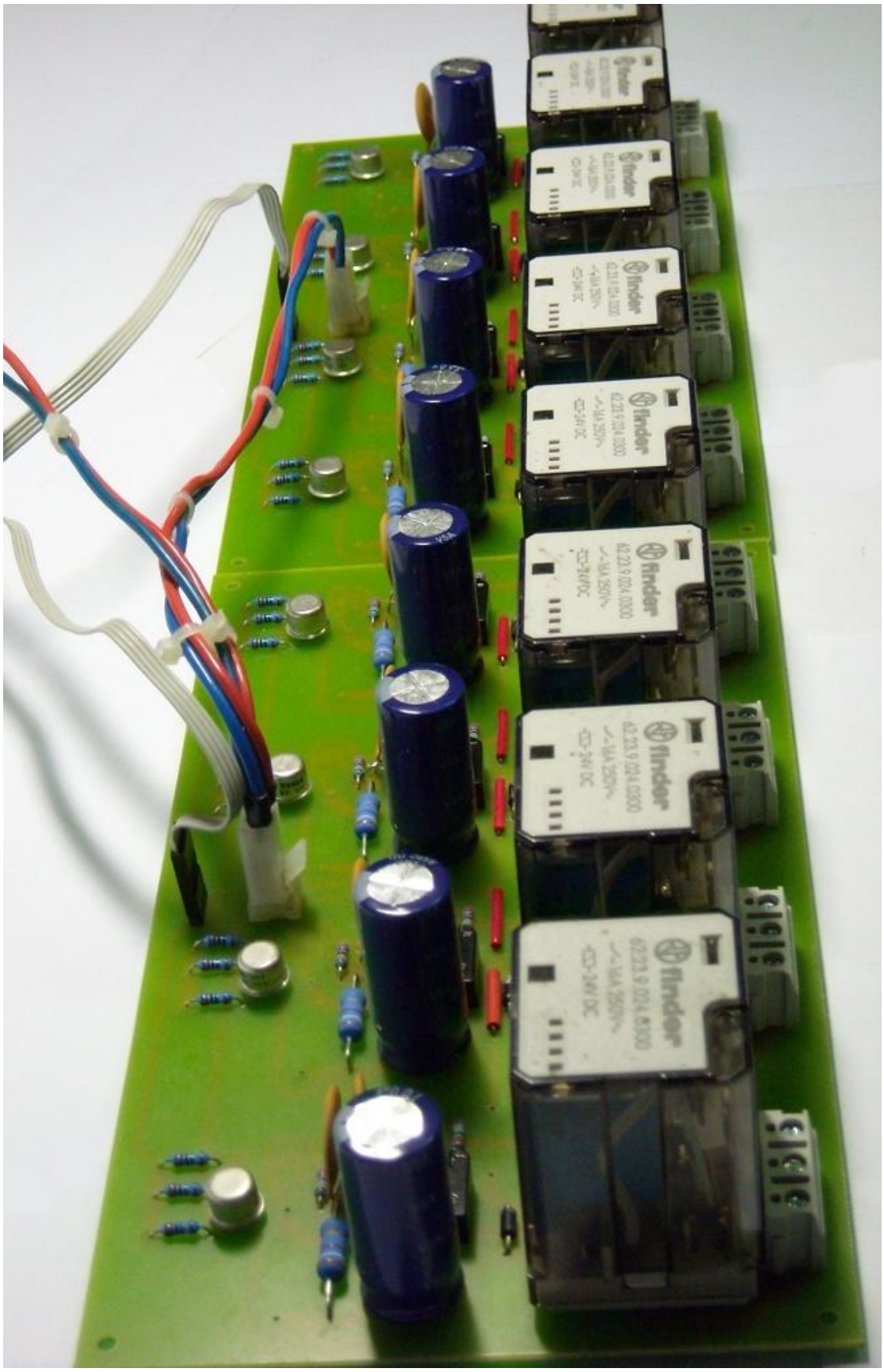
Napájecí obvod



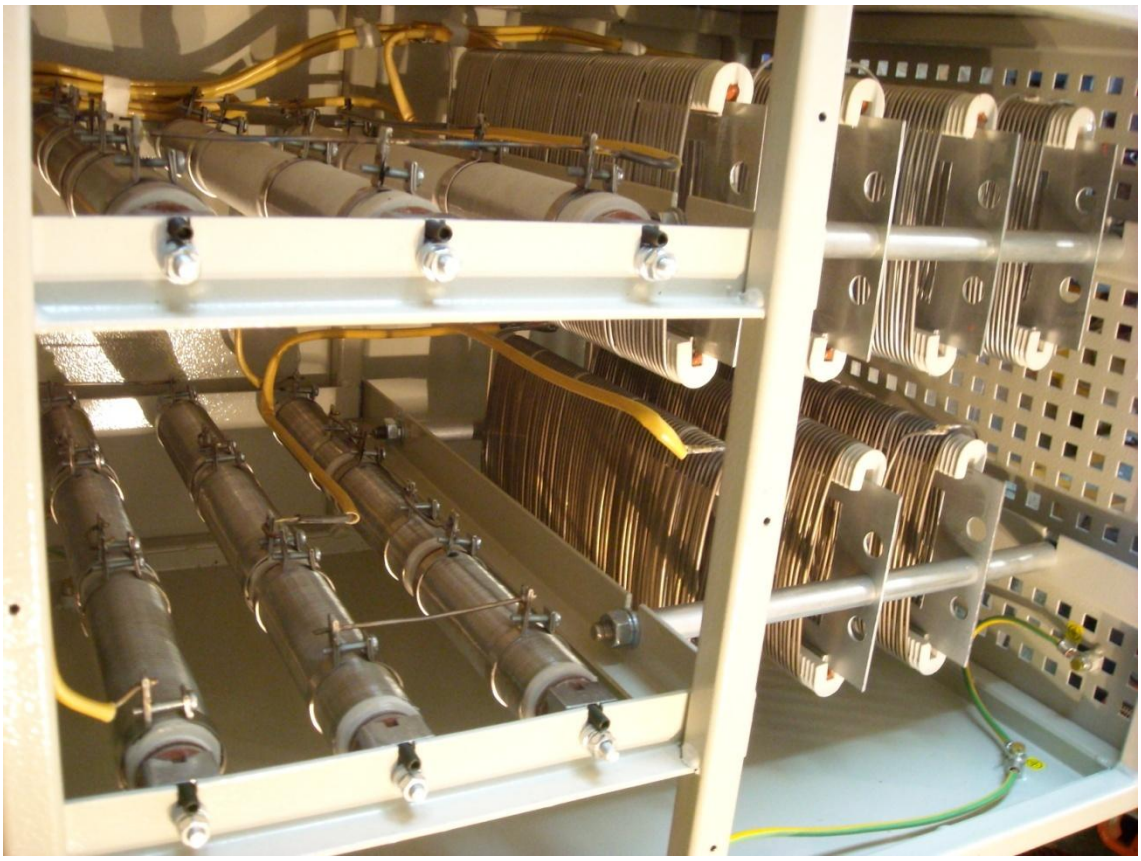
Výkonová (spínací) část obvodu



Výkonová část obvodu

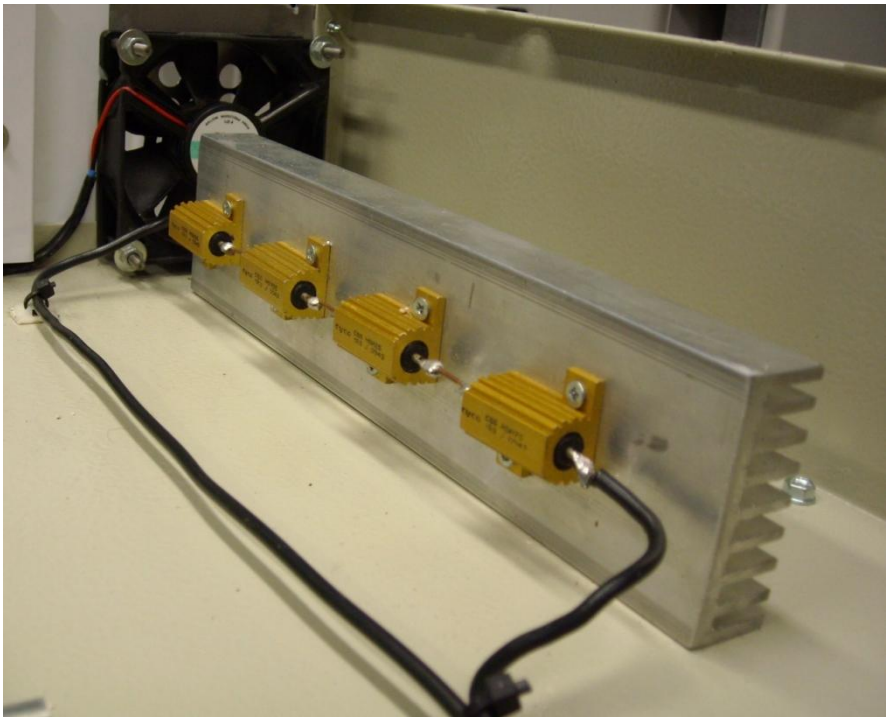


Výkonové drátové rezistory

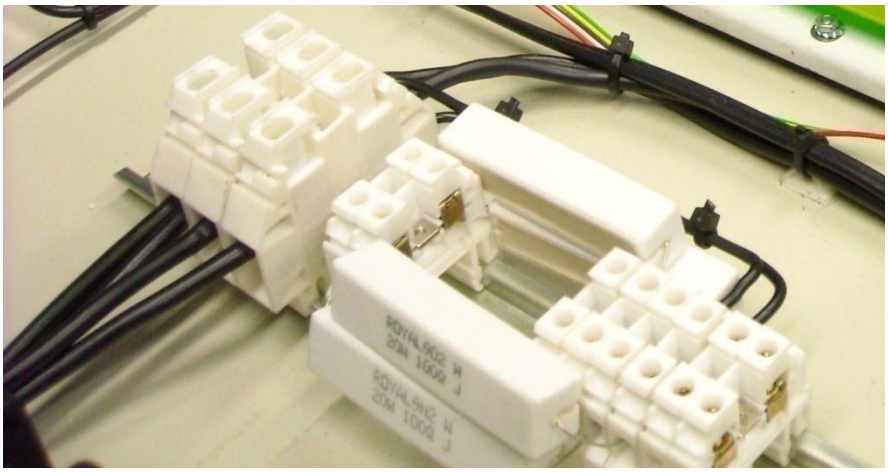


Chlazení rezistorů

Kompenzační rezistory 50-ti ohmového výkonového rezistoru (4×1 Ω)



Rezistory přidané k R1 (4 ×100 Ω)



Celkový pohled na zařízení

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
----------------------------------------------------------------------------------------------------------------------------
entity zatez is port(

clk             :               in std_logic;
rotary_a : in  std_logic;
rotary_b : in  std_logic;
tl_min      : in std_logic;
tl_max      : in std_logic;
USB         : in std_logic;
USB_D : in std_logic_vector (7 downto 0);
vystup_rele             : out std_logic_vector (7 downto 0):="00000000";
segment_1 : out std_logic_vector (6 downto 0):="1111111";
segment_2 : out std_logic_vector (6 downto 0):="1111111";
segment_3 : out std_logic_vector (6 downto 0):="1111111"

);
end zatez;
----------------------------------------------------------------------------------------------------------------------------
architecture Behavioral of zatez is

signal      citac1 : integer Range 0 to 9:=5;
signal      citac2 : integer Range 0 to 9:=5;
signal      citac3 : integer Range 0 to 9:=2;
signal      reg : std_logic_vector (7 downto 0):="11111111";
signal      pom_reg : std_logic_vector (7 downto 0):="11111111";
signal      reg_pan : std_logic_vector (7 downto 0):="11111111";
signal      value_dec : integer Range 0 to 255:=255;
            ----------------------------------------------------------------
 signal     rotary_a_in : std_logic;
 signal     rotary_b_in : std_logic;
 signal     rotary_in : std_logic_vector(1 downto 0);
 signal     rotary_q1 : std_logic;
 signal     rotary_q2 : std_logic;
 signal     delay_rotary_q1 : std_logic;
 signal     rotary_event : std_logic;
 signal     rotary_left : std_logic;
 signal     interrupt : std_logic_vector(1 downto 0):="11";
 signal     start_cas : std_logic;                          -- zpoždění výstupu relé
 signal     pocitat : std_logic;                    --spuštění čitače 2s.
 signal     pocitat_usb : std_logic;
 signal     enable_USB : std_logic;
 signal     reset : std_logic:='1';
signal      cas : integer range 0 to 120000;
signal      zapnout_vystup : std_logic:= '0';
signal      reset_zap : std_logic:='0';
signal      zacatek : std_logic;
signal      pom_usb : std_logic;
----------------------------------------------------------------------------------------------------------------------------
begin
--*************************************************************************************************
-----------------------------------------------------------------
--rozliseni smeru rotace enkoderu
-----------------------------------------------------------------
rotary_filter: process(clk)
  begin
    if clk'event and clk='1' then
            if enable_usb = '0' then
                                rotary_a_in <= rotary_a;
                                rotary_b_in <= rotary_b;
                                rotary_in <= rotary_b_in & rotary_a_in;
                                case rotary_in is
                                        when "00" => rotary_q1 <= '0';
                                                                rotary_q2 <= rotary_q2;
                                        when "01" => rotary_q1 <= rotary_q1;
                                                                rotary_q2 <= '0';
                                        when "10" => rotary_q1 <= rotary_q1;
                                                                rotary_q2 <= '1';
                                        when "11" => rotary_q1 <= '1';
                                                                rotary_q2 <= rotary_q2;
                                        when others => rotary_q1 <= rotary_q1;
                                                               rotary_q2 <= rotary_q2;
                                end case;
```

```
                        end if;
    end if;

  end process rotary_filter;
--*************************************************************************************************
direction: process(clk)
variable zapnout_pocitani : std_logic;
variable start_soucet : std_logic;

  begin
    if clk'event and clk='1' then
-----------------------------------------------------------------
-- osetreni USB
-----------------------------------------------------------------
            if USB = '1' then
                        pocitat_usb <= '1';
            else
                        enable_usb <= '0';
                        pocitat_usb <= '0';
            end if;
            if pom_usb /= enable_usb then
                        reset <= '1';
            end if;
            pom_usb <= enable_usb;
-----------------------------------------------------------------
-- ošetření enkoderu - zaregistruje pohyb enkodéru
-----------------------------------------------------------------
                        if enable_usb = '0' then
                        delay_rotary_q1 <= rotary_q1;
                        if rotary_q1='1' and delay_rotary_q1='0' then
                        rotary_event <= '1';
                                    zapnout_pocitani := '1';
                                    pocitat <= '0';
                        rotary_left <= rotary_q2;
                        else
                        rotary_event <= '0';
                                    if zapnout_pocitani = '1' then
                                    pocitat <= '1';
                                    zapnout_pocitani := '0';
                                    end if;
                        rotary_left <= rotary_left;
                        end if;
                        end if;--enable_usb
-----------------------------------------------------------------
-- casovac pro nastavení rele vystupu
-----------------------------------------------------------------
                                    if (pocitat = '1' or pocitat_usb = '1' or reset_zap = '0') then                    -- zahájit počítání

                                                cas <= cas + 1;
                                                if cas = 15000 then
                                                            start_soucet := '1';
                                                end if;
                                    if cas = 120000 then
                                                                                                                       ------Zpoždění 1,5 s
                                                            if pocitat = '1' or pocitat_usb = '1' then                 --
                                                                        if usb = '0' then
                                                                                    -- jestliže USB neni zapojeno přepíše se relé/výstup
                                                                        start_cas <= '1';
                                                                        -- zapnutí výstupu v konečném stavu (bez logického součtu)
                                                                                    start_soucet := '0';
                                                                        -- vypnutí logického součtu
                                                                        else
                                    -- jestliže je USB= '1' (připojeno), provede se enable USB - zapnutí PC režimu řízení zátěže
                                                                                    enable_usb <= '1';
                                                                        end if;
                                                            else
                                                            zapnout_vystup <= '1';
                                                            reset <= '1';
                                                            reset_zap <= '1';
                                                            end if;
                                                end if;
                                    else
                                                cas <= 0;
                                    end if;
```

```
-----------------------------------------------------------------
-- registry hodnot
-----------------------------------------------------------------
------------------------------------------------------------------ odčítání hodnot registrů při rotaci enkodéru
        if enable_usb = '0' then
    if rotary_event='1' then
                                    if rotary_left='1' then
                                                if value_dec > 0 then
        -- value_dec pouze pro funkci čitače
                                                        citac1 <= citac1 - 1 ;
        -- reg_pan registr pro vkládání hodnot z ovládacího panelu
                                                        reg_pan <= reg_pan - 1 ;
                                                        value_dec <= value_dec - 1;
                                                        if citac1 = 0 then
                                                                citac2 <= citac2 - 1 ;
                                                                citac1 <= 9;
                                                                        if citac2 = 0 then
                                                                                citac3 <= citac3 - 1 ;
                                                                                citac2 <= 9;
                                                                        end if;
                                                        end if;
                                                end if;

------------------------------------------------------------------ příčítání hodnot registrů
        else if (value_dec < 255) then
                                                        citac1 <= citac1 + 1 ;
                                                        reg_pan <= reg_pan + 1 ;
                                                        value_dec <= value_dec + 1 ;
                                                        IF (citac1 >= 9) THEN
                                                                citac1 <= 0;
                                                                citac2 <= citac2 + 1;
                                                                IF (citac2 >= 9) THEN
                                                                        citac2 <= 0;
                                                                        citac3 <= citac3 + 1;
                                                                        if (citac3 >= 2)then
                                                                                citac3 <= 0;
                                                                        end if;
                                                                END IF;
                                                        END IF;
                                                        end if;
                        end if;
                        end if;
-----------------------------------------------------------------
-- ošetření tlačítek Min. a Max. --
-----------------------------------------------------------------
        interrupt <= tl_min & tl_max;
                                                if interrupt = "01" then
        --tlačítko minimum
                                                        citac1 <= 0;
                                                        citac2 <= 0;
                                                        citac3 <= 0;
                                                        value_dec <= 0;
                                                        reg_pan <= "00000000";
                                                end if;
        --tlačítko maximum + reset             if (interrupt = "10" or reset = '1' or interrupt = "00") then
                                                        citac1 <= 5;
                                                        citac2 <= 5;
                                                        citac3 <= 2;
                                                        value_dec <= 255;
                                                        reg_pan <= "11111111";
                                                        reset <= '0';
                                                end if;
        end if;
                                                if (interrupt /= "11" or reset = '1' )then
        -- vypnutí zpoždění změny výstupu ( při stisku tlačítek se časovač neuplatňuje)
                                                        start_cas <= '1';
                                                        pocitat <= '0';
                                                        cas <= 0;
                                                end if;
-----------------------------------------------------------------
-- zápis do registru --
-----------------------------------------------------------------
                                    if enable_usb = '0' then
                                                if start_cas = '1' then
        -- zapis do reg po uplynutí nast. času
```

```
                                                REG <= reg_pan;
        -- do REG (registr pro relé/výstup) je uložen registr s hodnotami z ovládacího panelu
                                                start_cas <= '0';
                                        end if;
                                end if;
                                if enable_usb = '1' then
                                        REG <= USB_D;
        -- do REG je uložena hodnota z FIFO bufferu USB
                                end if;

-----------------------------------------------------------------
-- zápis na relé/výstup –ošetření relé logickým součtem—je třeba přidat časovač po změně FPGA nebo snížení takt. frekv
-----------------------------------------------------------------

                                if zapnout_vystup = '0'  then
        -- po zapnutí vystup relé = 0000..
                                                zacatek <= '1';
                                                vystup_rele <= "00000000";

                                else
                                        zacatek <= '0';
                                        if start_soucet = '1' then
                                                vystup_rele <= not (REG or pom_reg);
        -- logický součet se používá aby se při změně, nejdříve příslušná relé sepla a poté teprve příslušná relé vypla - jinak
dochází při změně odporu ke špičkovým hodnotám
                                                pom_reg <= REG;
        -- pouze pomocný registr
                                        else
                                                vystup_rele <= not REG;
        -- nastavení relé výstupu
                                        end if;
                                end if;
        end if;
  end process direction;
--*************************************************************************************************
        PROCESS (clk,citac1)
        -- Nejnižši digit
        BEGIN
        if (clk'event and clk='1') then
                if enable_usb = '0' then
                        if zacatek = '1' then
                                segment_1 <= "1111111";
                        else
                CASE citac1 IS
                                WHEN 0 =>  segment_1 <= "1111110";
                                WHEN 1 =>  segment_1 <= "0110000";
                                WHEN 2 =>  segment_1 <= "1101101";
                                WHEN 3 =>  segment_1 <= "1111001";
                                WHEN 4 =>  segment_1 <= "0110011";
                                WHEN 5 =>  segment_1 <= "1011011";
                                WHEN 6 =>  segment_1 <= "1011111";
                                WHEN 7 =>  segment_1 <= "1110000";
                                WHEN 8 =>  segment_1 <= "1111111";
                                WHEN 9 =>  segment_1 <= "1111011";
                                WHEN others =>  segment_1 <= "1011011";
                END CASE;
                end if;
                else
                segment_1 <= "1001110";
        -- písmeno C, je-li připojeno PC
                end if;
                end if;

        END PROCESS;
--*************************************************************************************************
        PROCESS (clk,citac2)

        BEGIN
        if (clk'event and clk='1') then
                if enable_usb = '0' then
                        if zacatek = '1' then
                                segment_2 <= "1111111";
                        else
                CASE citac2 IS
                                WHEN 0 =>  segment_2 <= "1111110";
```

```
                                        WHEN 1 =>  segment_2 <= "0110000";
                                        WHEN 2 =>  segment_2 <= "1101101";
                                        WHEN 3 =>  segment_2 <= "1111001";
                                        WHEN 4 =>  segment_2 <= "0110011";
                                        WHEN 5 =>  segment_2 <= "1011011";
                                        WHEN 6 =>  segment_2 <= "1011111";
                                        WHEN 7 =>  segment_2 <= "1110000";
                                        WHEN 8 =>  segment_2 <= "1111111";
                                        WHEN 9 =>  segment_2 <= "1111011";
                                        WHEN others =>  segment_2 <= "1011011";
                        END CASE;
                        end if;
                        else
                        segment_2 <= "1100111";
                -- písmeno P, je-li připojeno PC
                        end if;
                        end if;
                END PROCESS;
--*************************************************************************************************
                PROCESS (clk,citac3)

                BEGIN
                if (clk'event and clk='1') then
                        if enable_usb = '0' then
                                if zacatek = '1' then
                                        segment_3 <= "1111111";
                                else
                                        CASE citac3 IS
                                                WHEN 0 =>  segment_3 <= "1111110";
                                                WHEN 1 =>  segment_3 <= "0110000";
                                                WHEN 2 =>  segment_3 <= "1101101";
                                                WHEN 3 =>  segment_3 <= "1111001";
                                                WHEN 4 =>  segment_3 <= "0110011";
                                                WHEN 5 =>  segment_3 <= "1011011";
                                                WHEN 6 =>  segment_3 <= "1011111";
                                                WHEN 7 =>  segment_3 <= "1110000";
                                                WHEN 8 =>  segment_3 <= "1111111";
                                                WHEN 9 =>  segment_3 <= "1111011";
                                                WHEN others =>  segment_3 <= "1101101";
                                        END CASE;
                                end if;
                        else
                        segment_3 <= "0000000";
                        end if;
                        end if;
                END PROCESS;

end Behavioral;
```

Přílohy volně vložené

- Příloha F – Schéma zapojení řídicí jednotky
- Příloha G – Schéma zapojení výkonové části
- Příloha H – Schéma zapojení napájecího zdroje